博士論文

A Ceiling Damage Detection System Using Deep Learning Approach (Convolutional Neural Networks)

（深層学習（畳み込みニューラルネットワーク）による天井被害検出システム）

王　璞瑾

Contents

1. Introduction

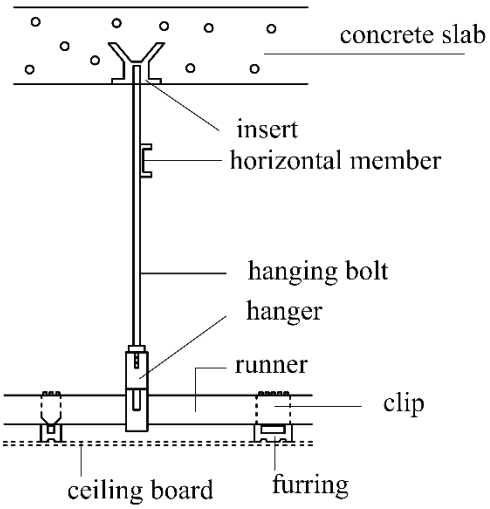
1.1 Backgrounds of ceiling damage detection

1.1.1 Ceilings and damages in ceilings

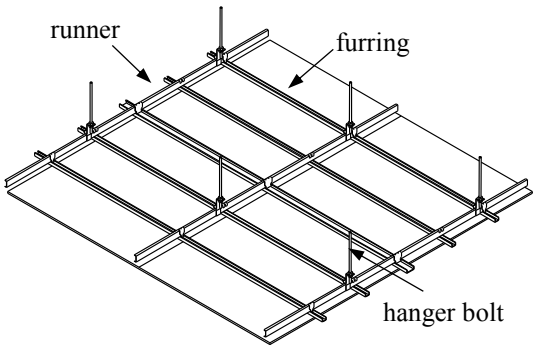
Structures have been resisting natural forces, like gravity, wind, moisture and outside loads ever since they are in construction to provide shelters and living space for human. To prevent life and property loss from structural failures, researchers and constructors have made enormous achievements in maintaining the safety of structural components under extreme circumstances. However, failure or falling of non-structural components are also dangerous to vulnerable human body. Fails of structural components are not always due to the traditional structural forces like earthquakes or winds that directly impose on them, but usually are caused by deteriorations and corrosions in the materials. Ceilings, one of the most widely used non-structural components, take a major role in interior design, thermal and acoustic environment control. Failures of ceilings are especially dangerous to human body [1-5].

Ceilings constitute the upper part of the interior space, work both aesthetically and structurally. Ceilings perform decoration function as non-structural components and bear load as structural components. Overhead mechanical, electrical and plumbing components are possible to be hidden behind the ceilings to serve the apparent requirements of interior design. Fire resistance, sound absorption / insulation, lightings, thermal insulation and other indoor environment performances require high-level design of ceilings. For all the requirements above, designers are prone to lower the priority or even forget to consider the structural safety in ceilings. Fig. 1.1 shows the system composition of suspended ceilings which are generally used in Japan and Fig. 1.2 shows the components of systematic ceilings, which are usually adopted in office buildings.

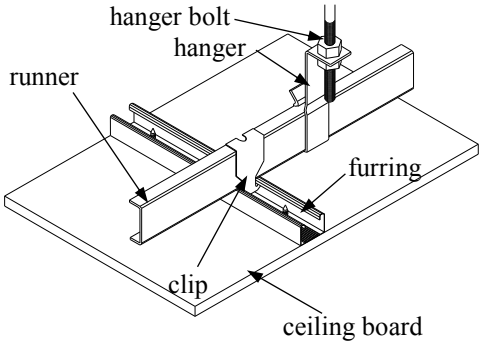
Designers and manufacturers usually pay much attention on the seismic resistance of ceilings [6]. However, fall of ceilings occurs not only under earthquake, but also occurs within daily life [7-11]. Ceilings interact with surroundings such as moisture, wind, temperature changes, rusting, leakage of rain, traffic vibration and aging in the materials. Ceilings of large space such as stations, public baths and natatoriums especially suffer from moisture, dew condensation and traffic vibration. In Fig. 1.3, examples of damages and fall of ceilings are shown.



(a) Sectional view



(b) View from the inner of ceilings



(c) Details of inner view

Fig. 1.1 Components of suspended ceilings

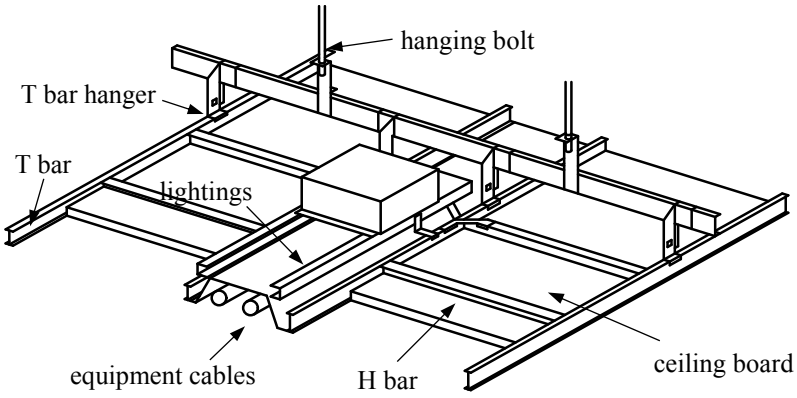


Fig. 1.2 Components of systematic ceilings

(a) Leaking of rain

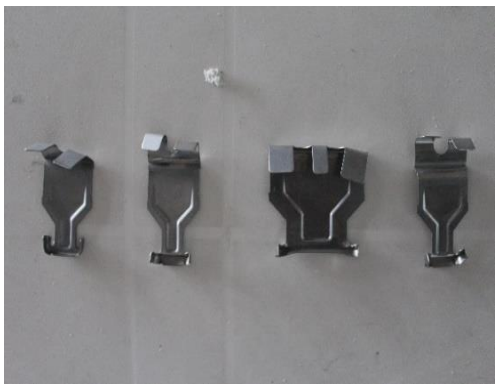
(b) Rust in ceiling frame

(c) Ceiling fall by dew condensation

(d) Ceiling board fall by earthquake

Fig. 1.3 Examples of damages in ceilings

There are four main forms to the fall of ceilings shown in Fig. 1.4: 1. Shape deformation of clips; 2. Slip off of screws; 3. Deformation of hangers; 4. Fall off of hanger bolts. The most frequent damage in ceilings is the shape deformation of clips, which is usually caused by earthquakes. The separation in the junctions is related to the aging of the ceilings. Condensation, leakage of rain, rusting and aging in the connections of ceiling boards and screws cause such kind of ceiling fall, which usually happens in spaces with high humidity. The dangerous aspect of the separation in the junctions is that such kind of ceilings fall occurs in a sudden, without any early warning to people under the falling ceilings.



(a) Shape deformation of clips

(b) Separation in the junction

(c) Deformation of hangers

(d) Fall of hanger bolts

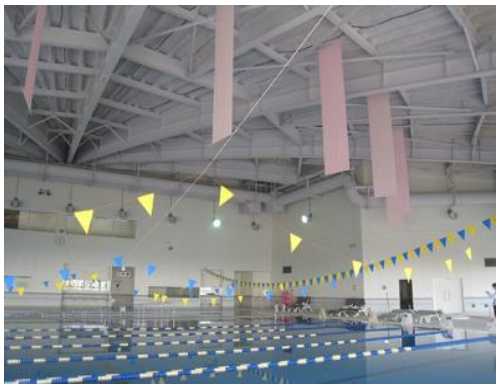
Fig. 1.4 Reasons to ceilings fall

Countermeasures to the fall of ceilings are: 1. Setting up of fall prevention net and wires, 2. Lightweight and softening ceilings, 3. Earthquake resistance enhancement, 4. Removal of the ceilings (shown in Fig. 1.5). These countermeasures have their own merits and demerits suited for different situations. They are possible to be in combination with each other to meet requirements of ceilings design.

(a) Fall prevention net and wires

(b) Lightweight and softening ceilings

(c) Earthquake resistance enhancement



(d) Removal of ceilings

Fig. 1.5 Countermeasures to the fall of ceilings

The prevention of ceilings from falling consists both “Life protection (solutions to prevent injuries and deaths due to ceilings fall)” and “Function maintenance (solutions to maintain the functions of ceilings like acoustic performance, heat insulation, aesthetic value and etc.)” [12]. The latter one is based on the realization of the former one. The most dangerous situations for falling ceilings are: 1. Many people gathering together, 2. Ceilings hung in high place, 3. Large area covered with ceilings. Buildings or architecture spaces with these three characters need special attention and safety evaluation regularly.

Fall of ceilings and other non-structural components causes problems:

1. Lives of people who are in the building are exposed to danger.

2. Long term impairment to spatial function of the building.

3. Occurrence not only under circumstance of earthquake but also at ordinary time.

4. The damages in ceilings are severer than those in structural components under the

same seismic degree.

The impediments among the solutions to these concerns include:

1. The deficiency of proper evaluation methods to the safety or danger degree of the ceilings installed on high place.

2. The lack of countermeasure methods to existing ceilings that may have been damaged.

3. The installation positions, forms and materials of the ceilings are determined by the appearance designers who are lack of education to the safety in ceilings.

4. The tendency of reinforcing the ceilings (non-structural components) in the same way to reinforce structural components that makes the ceilings much more heavier and much more damages if the ceilings fall.

5. The restoration of the ceilings to the state before accident happens also causes problems. It results in the recurrence of the same damages to the ceilings if another likely accident happens.

6. The absence of service life limit to the ceilings results in long-term service of the ceilings. This results in the fall of ceilings due to the deterioration of ceiling materials.

From the holistic perspective, safety and function preservation of ceilings are related to the height the ceilings installed and the materials of the ceilings. Appropriate danger degree evaluation to these aspects prevents fall of ceilings from happening and realizes the protection of lives. In the case of assumption that fall of ceiling occurs, a possible danger degree evaluation perspective is to evaluate the contact process between the ceilings and human body. Evaluation between the intensity of the contact and human body tolerance is possible to be converted to danger degree evaluations of ceilings [13-19]. Based on a series of tests that evaluates striking forces by different materials ceiling boards those fall from different heights on a simulated human head, relationships among the ceiling installation height, the ceiling materials and the final striking forces are established. Human body tolerance to these striking forces are compared to evaluate the danger degree of falling ceilings.

However, the fall of ceilings may happen under many different circumstances. The danger evaluation criteria that are based on the final impact force on human body may neglect the development of little damages in the ceilings. The little damages may

develop into big ones if there is no regular examination. Reliable solutions to detect damages in ceilings can prevent the problem of possible injuries to human from happening at the very beginning.

1.1.2 Damage detection of ceilings

Large span structures, like indoor stadiums in school, public buildings and hospitals have a potential to be shelters for residents when disasters like earthquakes and aftershocks occur in Japan[20]. These structures are designed to resist disasters without severe structural damages in columns or beams. Ceilings (including suspended ceilings, lighting equipment, inner / exterior finishing materials, etc.) are reported damaged during the Great East Japan Earthquake on 11th March 2011 even if they were constructed under the latest Japanese construction technical advice issued by the Ministry of Land, Infrastructure, Transport and Tourism of Japan (MLIT) [21]. In the guidebooks by Ministry of Education, Culture, Sports, Science and Technology - Japan (MEXT), non-structural components failures, ceilings for the most part, especially those that happen during aftershocks when people in the shelters, are reported to cause losses of lives and properties [22-24]. These guidebooks suggest inspection approach of nonstructural members in school facilities, especially ceilings. It is important to find any trace of abnormality to apply countermeasures at early stage before disaster happens.

In a Notification by the Ministry of Land, Infrastructure, Transport and Tourism of Japan (MLIT) [25], the inspection items of ceilings include both the outside part that directly faces to the room and the inner side where hangers are hidden. The inspection method is mainly on-site-inspection by human naked eye to find out if there is any damage, like floating in the boards, deflection, spalling, corrosion, loose, disengagement or deficiency. Binoculars telescope is to aid the inspection if necessary.

The inspection of ceilings begins with the collection of information of ceiling status in the purpose of protecting human lives. Then the function maintenance of ceilings requires the information of damages in ceilings [12]. The information relatives to not only the ceiling and ceiling foundation materials, but also to the type of the building, the location of the ceilings, suspended units, hanging facilities, columns and walls at the edges of ceilings. Table 1.1 shows the investigation items of ceilings when detecting damages to general utilized lightweight steel frame ceilings [12].

Table 1.1 An example of investigation items of ceilings		
Investigation item	**Contents**	**Details of investigation**
1. Structural type and form	Steel framed structure, steel framed reinforced concrete structure, reinforcement structure and other structural forms	Confirm the influence extent to ceilings by earthquake, wind and other external force.
2. Ceilings structural information	Height from ceilings to floor, ceilings and its foundation material, density of the materials, total area of the ceilings, shape of the ceilings, purpose of the room where the ceilings are	Damage evaluation if ceilings fall does occur
3. Ceilings working status	Moisture, water stains, wind pressure, structural shake	Influence on the live protection and functional maintenance
4. Earthquake resistance	The completeness of earthquake resistance components, performances of braces	Confirmation to the safety of the ceilings structure in earthquake condition
5. Facilities installed in ceilings	Equipment machines, inspection scaffold, audio equipment	Confirmation of the influence to the ceilings foundation and the existence of fall-prevention
6. Surroundings of ceilings	Walls, columns and other structures connected or next to ceilings	Evaluation the impact if earthquake occurs

Non-destructive ceiling examination systems to existing historical buildings applying methods like ultrasonic echo technique, ground radar and measurement by reinforcement scanner are also reported [26]. Different tasks of examination need different methods. A combination of these methods would reach better results. These methods require that the testing personnel have tremendous practices and experiences because ceilings are constructed by diverse materials and forms. In the inspection and diagnosis for gypsum plasters ceilings, an expert system was developed [27]. This system includes a defect classification and probable causes of these defects, which is also based on on-site inspection of human naked eyes. It provides an easy method to analysis possible causes to the defect and to find proper solutions to that.

There are also attempts to free professionals or people who follow the long inspection list from on-site inspections. Smart sensor board and inspection robots are exploited to evaluate the ceiling condition, detect the location and condition of the damage in another report [28]. This method can detect more details of the ceilings on the inside

part where even professionals cannot see only by on-site-inspection. However, this method is only suit for specific ceilings. Another solution using a series of algorithms named Simultaneous localization and mapping (SLAM) consists of the construction of a model of the surrounding environment and a robot moving in it has made astonishing progress in the last thirty years [29]. The robot understands the topology of the environment and builds information model (map) to perform actions. Solutions to estimate the pose of the robot is to build a ceiling-feature map by using an upward looking monocular camera [30, 31]. Ceiling feature extraction methods in these solutions are heuristic algorithms to detecting damages in the ceilings.

All the algorithms and solutions of recognizing damaged ceilings mentioned above have common defects: either they need considerable experiences and work by the observer (human) or they are not suitable to the complex working conditions when complex algorithms are applied. A robust, reliable and diversity-adapted solution of damage detection is in need.

1.1.3 Damage detection to structures and the positioning of this thesis

In structural engineering and civil engineering, the damage detection to the target structures overlaps with the structural health monitoring (SHM) [32-34]. The damage detection uses many information monitored from the structure such as the vibration [35, 36], the frequency [37], the temperature [38], the acoustic performance [39, 40], the damping [41] and etc. The damage detection and SHM usually contain two main parts: 1. the collection of the data by sensors and 2. algorithms processing the data to make judgement. Researchers have been developing new sensors to detect different physical signals generated or reflected from the objective structures, and more algorithms to process these signals to extract more information about the target structure. Algorithms such as principal components analysis (PCA) [42-44], genetic algorithm (GA) [45, 46], support vector machine (SVM) [47] are applied in structural damage detection.

Among the algorithms for structural damage detection, an early research using neural networks for structural damage detection is developed in 1992 [48]. In this article, a neural network for recognizing the behavior of undamaged structures and damaged structures is trained using the self-organization and learning capabilities of neural networks. The input to the neural network is the frequency respond of a structure, which is the computed acceleration time histories of the structure using Fourier spectra of the acceleration time histories. The output is a number ranging from 0 to 1, indicating the damage state of the input. Even though there are only 42 inputs, the whole process and the idea behind this research contains all the necessary elements in neural network application. Other researches using neural networks for structural damage detection

follow the same process as data processing, model building and training, model validation [49-52]. However, these researches all use the vibration data for input, but fail to attempt using other monitoring data.

Although monitor images or videos are easy to acquire, damage detection and SHM algorithms using images or videos of the target structures are still few until recent years. Two possible reasons are that the analysis of images and videos will consume a big amount of calculation resource and the algorithms for image and video processing are still under research. There are algorithms to detect cracks in concrete which are possible to be modified to detect cracks in ceiling panels. Edge detection techniques find edges among pixels in gray level where contrast is over the threshold [53]. Four edge detection techniques are compared to finding crack in bridges [54]. Edge detection techniques need much parameter adjustment to reach the best results. Another image processing method for detecting concrete surface cracks is multiple sequential image filtering [55]. This method can accurately detect cracks in images recorded in various conditions and even quantify the widths of the detected cracks from the spatial derivatives of brightness patterns. A semi-automatic, texture analysis approach to detect and classify ageing infrastructural elements, using enhanced texture segmentation can fit the variations in different damage forms, lighting conditions, viewing angles, and image resolutions [56]. There is also an attempt to detect defects in reinforced concretes using the method named random neural architectures [57]. It is a machine learning method that train a model using data collected from real world. However, the complexness of the model is weak. A common defect of these image-based algorithms is that they can only detect simple damages such as cracks on the surface of concrete and rust on the surface of steels. More divers and robust algorithms for structural damage detection are still in need.

Deep learning connects the neural networks with the image analysis [58]. With the development of computer hardware and deep learning software infrastructures, there are many applications using deep learning for structural damage detection. Defect detection of reinforced concrete using random neural architectures provides a non-invasive technique to the structures [57]. Another concrete crack detection using deep learning and convolutional networks can detect and extract cracks on the surface of concrete [59]. In the field of steel structure surface detection, convolutional neural networks is also applicable [60]. Computer aided solutions to detect cracks or damages in structures using deep learning methods [61, 62]. Although they claim that they have reached very satisfying accuracies, the detectable damages are usually limited to cracks on the surface of concrete or corrosions on the steel, which have very common features that are easy for non-deep-learning algorithms to grasp. The real power of deep learning is far from being exerted in structural engineering. Furthermore, there have been no research in structural engineering or civil engineering on the mechanisms of the

artificial models for structural damage detection. Almost all the researches follow the same route: 1. Generate a database; 2. Train a neural network model until to a high accuracy; 3. Test the model with other data; 4. Claim that the model has good performance. In Table 1.2, a brief description to structural damage detection approaches using neural networks or deep learning is displayed. The most common defect is that none of them researched on the inner mechanisms of the neural network models.

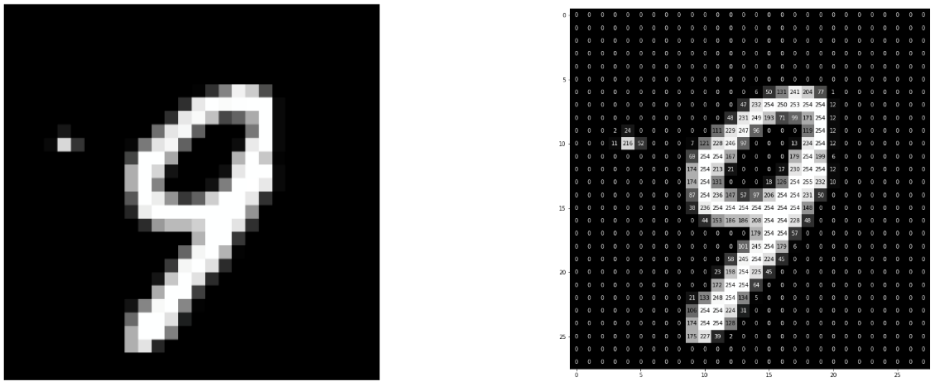
Table 1.2 Structural damage detection approaches using neural networks or deep learning

Researcher	Deep learning method	Year	Detectable damages	Advantage	Disadvantage
X. Wu, J. Ghaboussi, and J. Garrett Jr. [48]	Neural network	1992	Overall damage evaluation	Early application of neural network in structural damage detection	Too simple
P. Pandey and S. Barai. [49]	Neural network	1995	Damage in bridges	Attempts to apply neural network in bridge monitoring	Too simple / limitation in bridge form variations
C. Zang and M. Imregun. [50]	Neural network & PCA	2001	Frequency response functions	Evaluation to the structural status	Cannot locate the damage position
X. Fang, H. Luo, and J. Tang. [52]	Neural network	2005	Frequency response functions	Tune the model with learning rate improvement	Too simple algorithm adjustment
J.B. Butcher, et al. [57]	Random Neural Networks	2018	Concrete surface defects	Reduction of data collection time	Cannot locate damage region
Y.-J. Cha, W. Choi, and O. Büyüköztürk [59]	CNN	2017	Concrete cracks	High accuracy	The detectable damage is only concrete cracks
D. Soukup and R. Huber-Mork. [60]	CNN	2014	Steel surface damage	Vision-based	Too small images and too few detectable damage forms
Y.z. Lin, Z.h. Nie, and H.w. Ma. [61]	CNN	2017	Frequency response functions (FRFs) and vibration modes	Considering the visualizations to the hidden layers	Limited to beam components
Y.J. Cha, et al. [62]	Faster R-CNN	2017	Concrete crack, steel corrosion with two levels, bolt corrosion, and steel delamination	Detecting the damages in one-run	The detectable damages are still strong in characteristics

In this thesis, building and training a convolutional neural network model for ceiling damage evaluation is the first step. In fact, the intact forms and damaged forms in ceilings are much more various than cracks in the concrete, deep learning can exert its strengths in these complicated situations. Secondly, investigations to the model are the core in this thesis. Investigations are mainly performed by visualizations of the model to human interpretable images. Through the visualizations of the model, the ultimate objective of this thesis: the ceiling damage detection function is accomplished as well. Thirdly, a ceiling damage detection system involving in the user for interactivities is devised and tested. Finally, transfer learning is introduced to build more powerful CNN models.

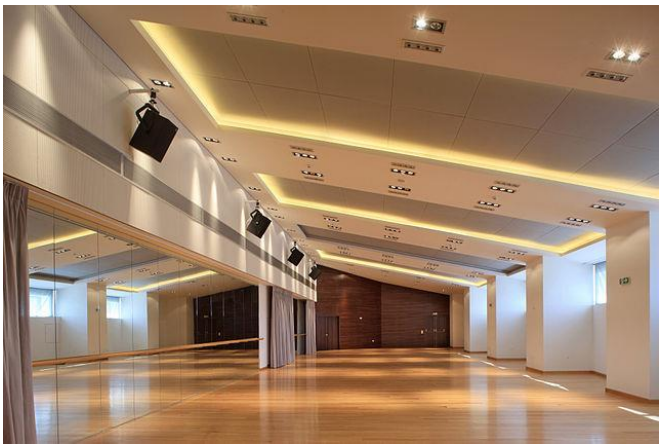
1.2 Image processing algorithms and machine learning

Image processing refers to the mathematical alternations to a digital image. The simplest digital image may be defined as a function $f(x, y)$, where x and y are the coordinates of a spatial plane (two-dimension), numerical value of $f(x, y)$ is the intensity / brightness / gray level of the pixel at the point (x, y) (usually ranges from 0 to 255). The pixels containing both spatial and intensity information constitute a whole digital image. One digital image is regarded as a series of numbers, or a matrix, to the computer [63]. Fig. 1.6 shows an image of the number nine (28 by 28 pixels of one channel) and its details in matrix. RGB channels of an image are shown in Fig. 1.7. Different channels are emphasizing different features of the original picture. Applying image processing algorithms to an image is altering the matrix in the image.

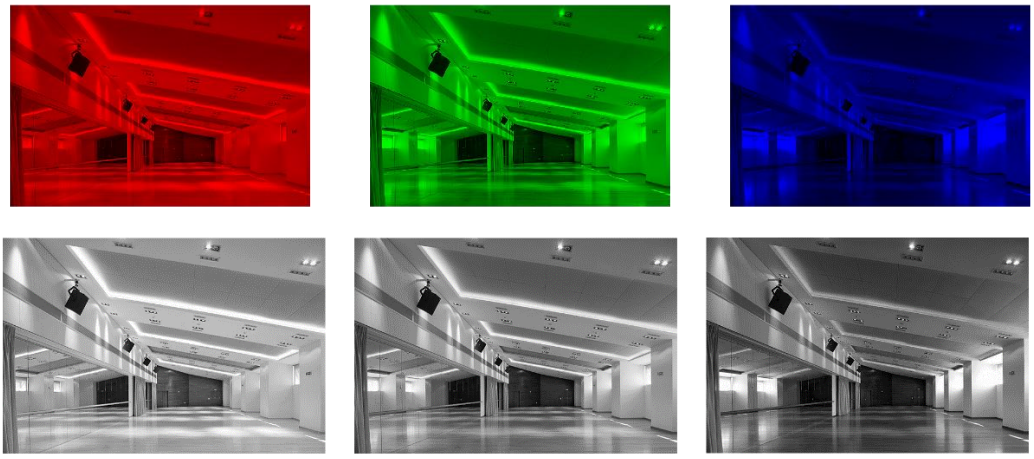


Original image Image interpreted by computer

Fig. 1.6 View of an image in detail (one channel)



Original image
("Badminton Theater Rehearsal Room",
by Fvonglower, is licensed under CC-BY-SA-3.0, resolution: 640×427×3)



RGB channels to the original picture
Fig. 1.7 RGB channels constitute one color picture

1.2.1 Image processing algorithms

Generally, any alternations applied to the original digital image can be named under image processing algorithms. Changes of shape, distortion, brightness, color are basic alternation algorithms (show in Fig. 1.8). These adjustments are easy to apply and understand, but they are too primitive to accomplish complex tasks in image processing.

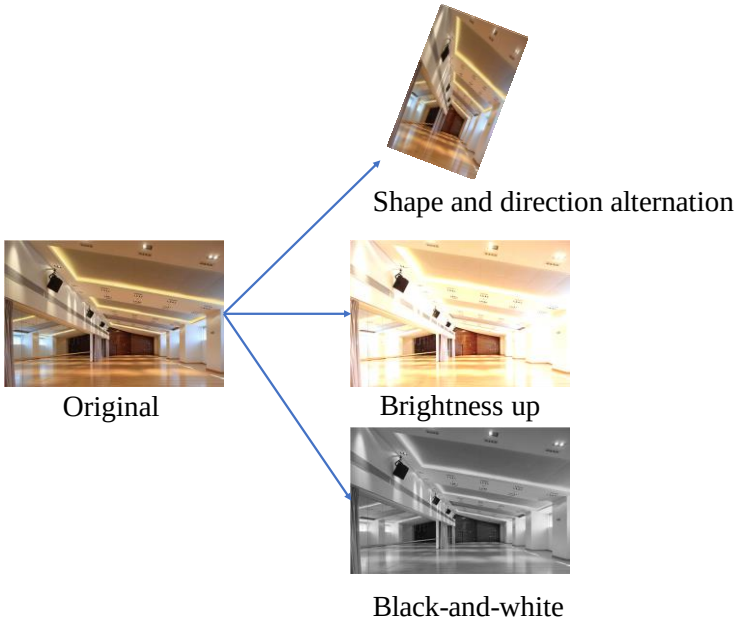
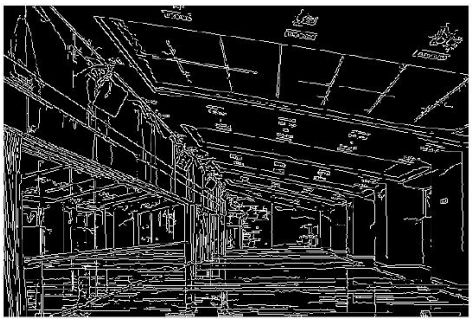


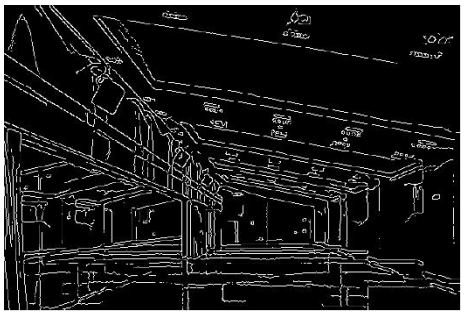
Fig. 1.8 Adjustments to an image

In the domain of image processing to ceiling images, there are too many varieties of ceiling shapes and materials, which are hard to interpret by the computer. The goal of Canny Edge Detection is to identify the boundaries of an object in an image [64]. Firstly, transfer the image into a grayscale image. Secondly, calculate the gradients in the image. The gradient is defined by how different the values are in adjacent pixels in the image. Each pixel in the gradient image corresponds to the strength of the gradient at this point. The edges of an object can be traced out by following the strongest gradients. There are three parameters to adjust in Canny Edge Detection, the low threshold, the high threshold and kernel size, which are adapted to detect edges of different objects (Fig. 1.9). The disadvantages of Canny Edge Detection are: 1. the adjustments to the parameters require experienced human; 2. Shadows by complicated light environment invalidate the edge detection process.

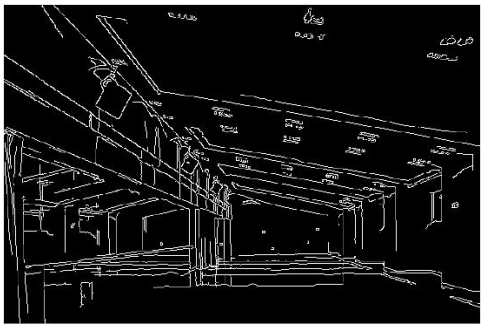
Original (grayscale)



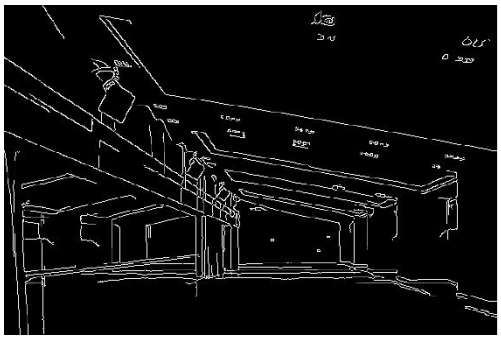
k=3, l=10, h=50



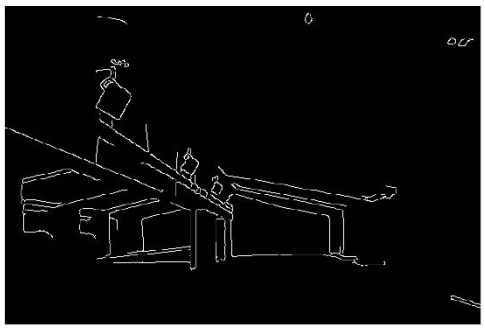
k=9, l=10, h=50



k=3, l=50, h=200



k=5, l=50, h=200

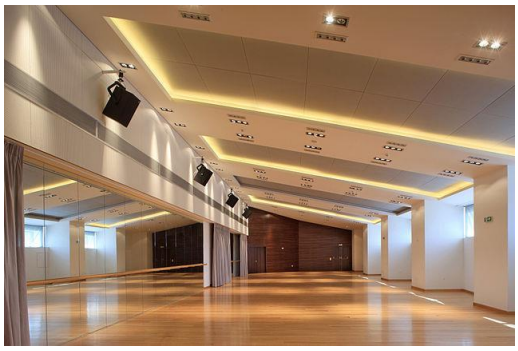


k=9, l=50, h=200

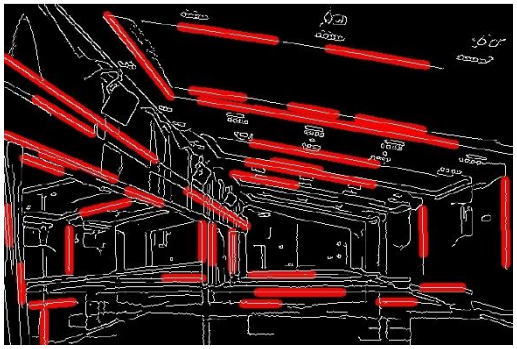
k: kernel size, l: low threshold, h: high threshold

Fig. 1.9 Canny Edge Detection for ceilings

Hough transform, devised by Paul Hough, can perform transformations of one line in a two-dimension image (image space) into one point in a Hough space, and a line in a Hough space back to one point in the image space [65]. To find specified lines (length and tilt) in the image space, the intersection point among intersecting lines in the Hough space can be calculated to determine the specified lines in the image space. To detect ceilings in an image, shapes of ceilings can be set by adjusting parameters of Hough transform. Fig. 1.10 shows the input and output by Hough transform.



Input



Output (red lines are recognized by Hough Transform)

Fig. 1.10 Hough Transform to Detect Ceilings

The algorithms above can be applied to find out the most important lines and shapes that compromise the ceilings and other components. There are also many algorithms that are good at processing colors, shapes, even recognize human faces. However, these algorithms are too specialized in their own fields and are hard to composite to meet the requirements of a complex mission in detecting and evaluating ceiling damages.

1.2.2 Machine learning

Computers are designed to calculate, not for perception of the world around them. They need to be programmed to finish tasks designated by human. Algorithms are developed to solve one task or a series of tasks which can be divided into small ones using explicit (a finite amount of space and time) specifications. In other words, computers are passive to the outside world and need to be programmed to do tasks. However, machine learning makes computers learn from experiences and weakens boundaries between humans and computers [66-68] . Machine learning has been applied in many fields in which only human was capable of before, like image recognition, voice translation, fraud detection in bank systems, spam detection, playing GO and driving cars. Past experiences need to be translated into digitalized information as input for computers, which also can be called “data”. Machine learning does not specify every line of code to solve a problem but builds a model which takes in data (past experiences) as input to optimize the performance of the model using pre-defined metrics. The “learning” process occurs in the optimizing the parameters in the model. A “trained” model is generated when the metrics reveal optimization and the model is ready to make predictions to new data.

Due to the properties that the models / algorithms of machine learning “learn” from data, machine learning also overlaps with artificial intelligence (AI) [68]. Machine learning models / algorithms are composed with different architectures but have something in common: they optimize millions or hundreds of millions of parameters in

them in the train phase, namely the learning process. Machine learning is especially good at judgement, which is also named classification. For example, machine learning can predict if a student can be admitted into a school base on the student status like grades and social activities. Machine learning can also segment customer groups based only on their consumption patterns. Table 1.3 introduces different kinds of machine learning and their properties. Machine learning is a series of algorithms with versatile capacities to be applied in our real life. The deep learning, a branch of machine learning, is reaching one and another the best state of art results in recent years. Almost all kinds of digital information can be processed and learnt by deep learning and some of them have already beaten human level results.

Table 1.3 Machine learning categories

Field of machine learning	**Characteristics and application**
1. Supervised learning	Given a series of input *X* variables and their output *Y* variables, build and train a model that reflects $Y = f(X)$, the model is good enough when given a new input *x*, it can precisely predict the output by calculating $y=f(x)$. "Supervised" refers to the existence of both input *X* variables and output *Y* variables when training the model. The *Y* variables supervise how the model learns like a teacher. The learning process stops when the performance of the model prediction is acceptable. Supervised learning is widely used in building judgement systems such as finding spam mails, voice recognition and recommendation products.
2. Unsupervised learning	When only input *X* variables are given, investigating and finding underlying structures of the *X* variables is unsupervised learning. Building a model that segments the whole input *X* variables into interpretable clusters is the goal of unsupervised learning. There is no correct answer (output *Y* variables) or no teacher to the input data, so such machine learning is called unsupervised learning. Unsupervised learning is useful in market segmentation of customers whose consumption customs are acquired by sellers. It is useful in dividing a series of experimental results to find out what condition affects the results most.
3. Semi-Supervised learning	When abundant input *X* variables but only a few output *Y* variables are acquired, such machine learning is semi-supervised learning. It is very common in real world machine learning problems because the output *Y* variables are labeled by human, especially by experts. The

	labeled data of output *Y* variables are expensive to acquire while the input data of input *X* variables are easily to acquire. A very common example of semi-supervised learning is diagnosis to a specified and relatively rare disease. There are many inputs (symptoms of illness) but very few outputs (confirmation to the specified disease).
4. Reinforcement learning	Reinforcement learning is different from finding a model that best suits the input *X* variables and output *Y* variables or segmenting a bunch of input *X* variables into clusters. Reinforcement learning makes an agent (a player controlled by the algorithm) who learns by itself in a given environment (a world defined by a series of rules). For example, a reinforcement learning algorithm controls a mouse in a defined maze to find a piece of cheese by learning the structure of the maze. Reinforcement learning can learn by itself to find the shortest way to the cheese by exploring the maze. Reinforcement learning is so powerful that it is possible to apply in all tasks in the real world.
5. Deep learning	Deep learning is also in the supervised learning frame. It learns with the optimization process which each of multiple processing layers calculates input from its previous layer and pass its output to its latter layer. Each layer abstracts the information gradually until the whole model best fits the problem. Deep learning could be applied in any field that input can be digitalized. In speech recognition, visual object recognition, semantic analysis and many other domains, deep learning is making astonishing achievements.

There are no absolute boundaries among algorithms of machine learning. By combination and reasonable arrangement of these algorithms, many problems can be solved elegantly by AI. For example, AlphaGo Zero and AlphaGo [69, 70] using deep reinforcement learning and other machine learning algorithms have beaten human in Go competitions which was considered impossible in classical algorithms. In recent years, people are pushing the boundaries of machine learning more and more further than before. Imagination and practice make machine learning shine in the future.

1.3 Applying machine learning to ceiling damage detection

1.3.1 Requirements to the deep learning model

The machine learning algorithms have their merits in analyzing problems that were very difficult for computers before. Classification of objects were very hard for computer to handle with because there are too many features in even only one object. However, the emergence of the machine learning, especially the deep learning, broadens the frontiers of artificial intelligence. The deep learning method can do multilayer abstractions to the inputs and their labels by optimizing weights in the deep learning model. The optimizing process can be deemed as the process of learning, although the real mechanism of human learning process is still under research. Fig. 1.11 shows the flow chart of how to generate a qualified CNN model for ceiling damage evaluation.

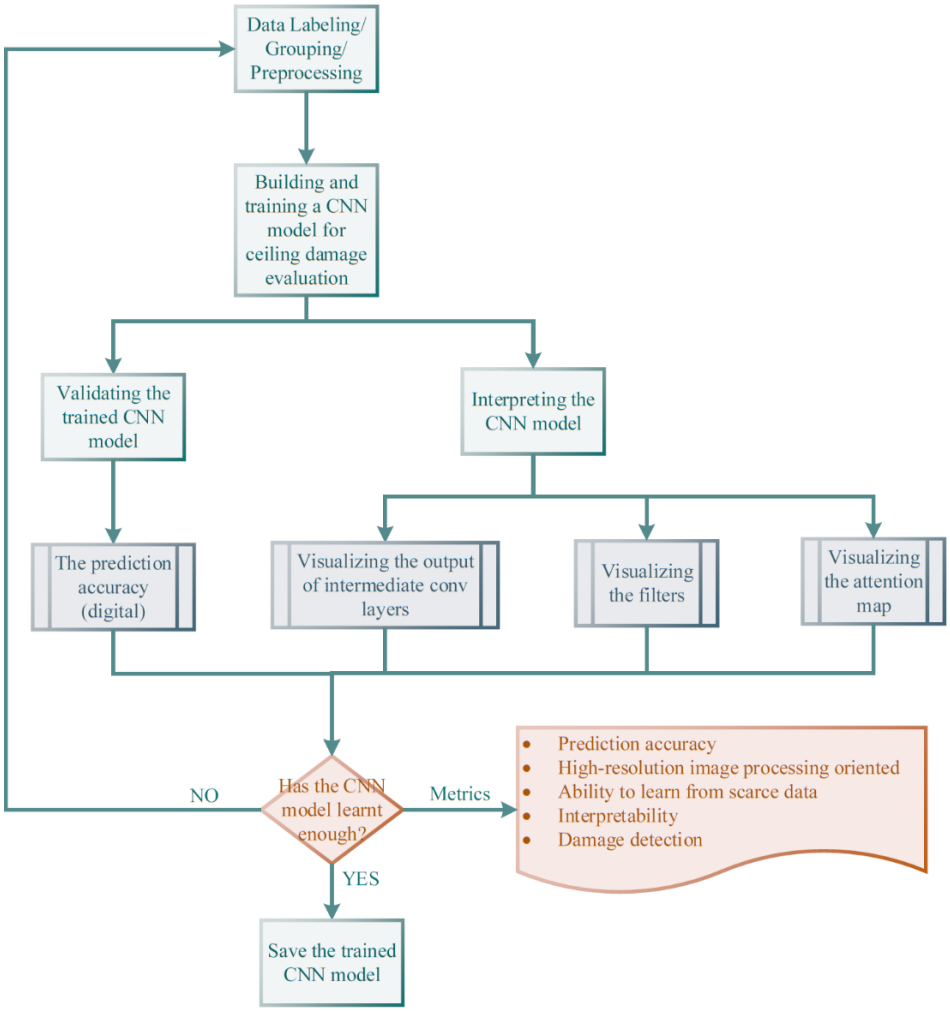


Fig. 1.11 Flow chart of building and training a CNN model

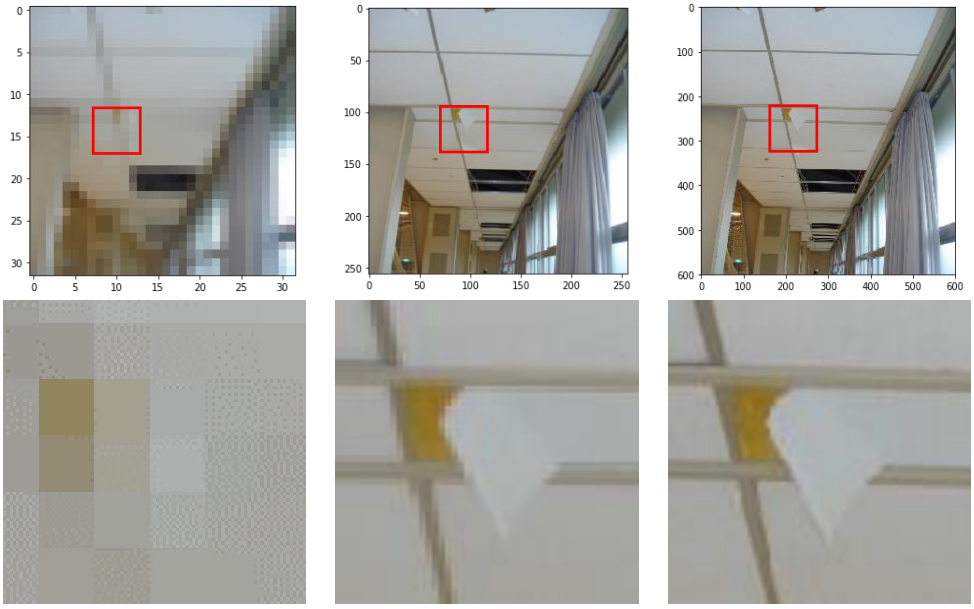
Noticing the metrics to decide if a CNN model has learnt enough in Fig. 1.11, these metrics are the requirements to the CNN model as well. To apply deep learning in ceiling damage detection, the deep learning model should meet the following **requirements:**

(1) Accuracy: The predictions made by the deep learning model are digital numbers that represent the possibility of the ceilings to be damaged. The predictions are required to be accurate enough to provide strong guidance to human. Deep learning (especially the CNN architecture) is good at classification. It can classify the images of different objects to a very high accuracy [71]. The ceiling damage detection task is different from the traditional classification task. The features in the images of ceilings are various and hard to directly classify. Moreover, a damaged ceiling image and an intact one may contain the same features that cover over 80% area in the image (shown in Fig. 1.12). The deep learning model are required to grasp the most important intact and damaged features that may only take a very small proportion to the whole image in area.

(a) Intact (b) Damaged

Fig. 1.12 Images sharing common features with different labels

(2) High-resolution image processing oriented: There are many possible damaged and intact forms in ceilings, the model must learn enough from the dataset. The model should be able to process relatively high-resolution images that contain more details in the ceilings. Any possible signs of damages in the ceilings should be noticed, even if they are tiny. Fig. 1.13 shows different resolutions to the same ceiling image.



(a) 32x32 (b) 256x256 (c) 600x600

Fig. 1.13 Different resolutions to the same image

(3) Ability to learn from scarce data: Most deep learning models are trained on massive data (thousands or millions of images for classification task). However, when applying deep learning methods to ceiling damage evaluation and detection, the images that well representing the features of the ceilings, especially the damaged ones, are quite few (2,000 ceiling images in total). The deep learning model should be modified to learn enough from the training data even if the training data is in scarcity.

(4) Interpretability: The accuracy of the predictions is important, but not everything. People used to pursue the accuracies by aggressively increasing the complexity of the deep learning models with the sacrifice of interpretability of the models. Interpretability refers to the understandings and faith of human to the deep learning model (shown in Fig. 1.14). Although there are many emerging new methods and architectures to improve the prediction accuracies in deep learning, there is not much research on the interpretations of the deep learning models. To some extent, the deep learning is still a black box to human [72]. The deep learning algorithms work, but we do not know why they work. The mechanisms in the deep learning model should be understandable or at least evaluable.

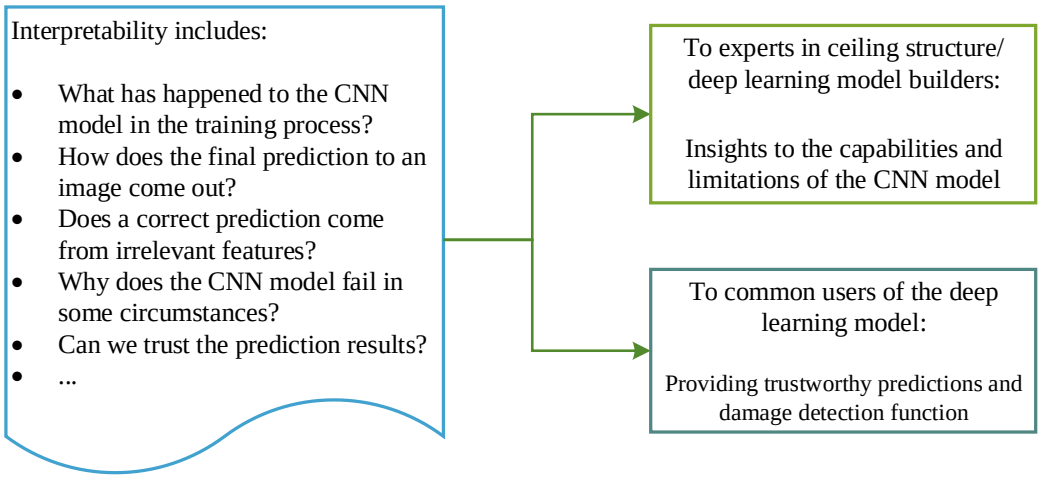


Fig. 1.14 Interpretability

(5) Damage detection function: The damage detection is the ultimate objective in this research, it is also a challenge to the deep learning model as well. The damage detection system using deep learning should provide possible damaged regions in the image to the user.

1.3.2 Solutions to the requirements

To meet the requirements above, the knowledge of deep learning and ceiling damage detection should be deeply intertwined to build and train the deep learning model. **Solutions** that modify the deep learning models to meet the requirements in the ceiling damage detection task are:

(1) Realizing the high accuracy in the predictions:

There are many knobs to tune in deep learning, the most widely accepted evaluation standard to a deep learning model is its prediction accuracy. Researchers have been struggling to improve their final prediction accuracy since the first day of deep learning. In this thesis, many attempts and trials like tuning the architecture of the CNN model, data pre-process and training process are performed to get a final relatively satisfactory prediction accuracy.

(2) High-resolution image recognition solution:

The resolution of 400×600×3 is chosen for the images in training data, which is a balance of large information capacity in one image and the computing resource consumption. The architecture of the CNN model is also adjusted to perform gradual abstractions in the high-resolution images.

(3) Data augmentation to generate more data for training:

The lack of data leads to overfitting, which means that the model mechanically remembers irrelevant features in the training data and make predictions by recognizing these irrelevant features. According to the unique property of convolutional neural networks, translation invariance, alternations to the original images will not change the contents in them and will generate more data for training and testing. Data augmentation algorithms are adopted to alleviate data scarcity.

(4) Visualizing the trained CNN model to make it interpretable:

Different levels of visualizations to the trained CNN models are used: (1) Visualizing the middle convolutional layers to find what regions in the input image most activate the filters in the convolutional layers; (2) Visualizing the patterns that most activate a specific filter in a convolutional layer to investigate the different degrees of abstractions in the convolutional layers and visualize what the CNN model has learnt; (3) Highlighting the pixels that contribute most to the final predictions using saliency map in the calculation of backpropagation, which is a relatively aggressively coarse visualization method; (4) Visualizing the attention maps in the trained CNN model using gradient-weighted Class Activation Mapping (Grad-CAM) method, which is a more exquisite and precise visualization method. By using these visualizing methods, a CNN model is interpretable to human.

(5) Visualizing the attention maps to accomplish the damage detection function:

Object detection is another popular deep learning research field that attracts lots of clever brains, many methods are also developed for the object detection task. In this thesis, visualizations of the attention maps to the trained CNN model provide the solution to the damage detection task since the trained CNN model has been properly trained and interpreted.

1.3.3 New proposals

Although the history of deep learning dates back to a few decades ago, the real rise of deep learning occurs in the recent no more than ten years, within which the most fruitful achievements are in recent five years. The tentacles of deep learning have touched almost every field in which the data can be normalized. Fig. 1.15 shows the new proposals to fulfill the requirements for the ultimate objective and the significances of the research.

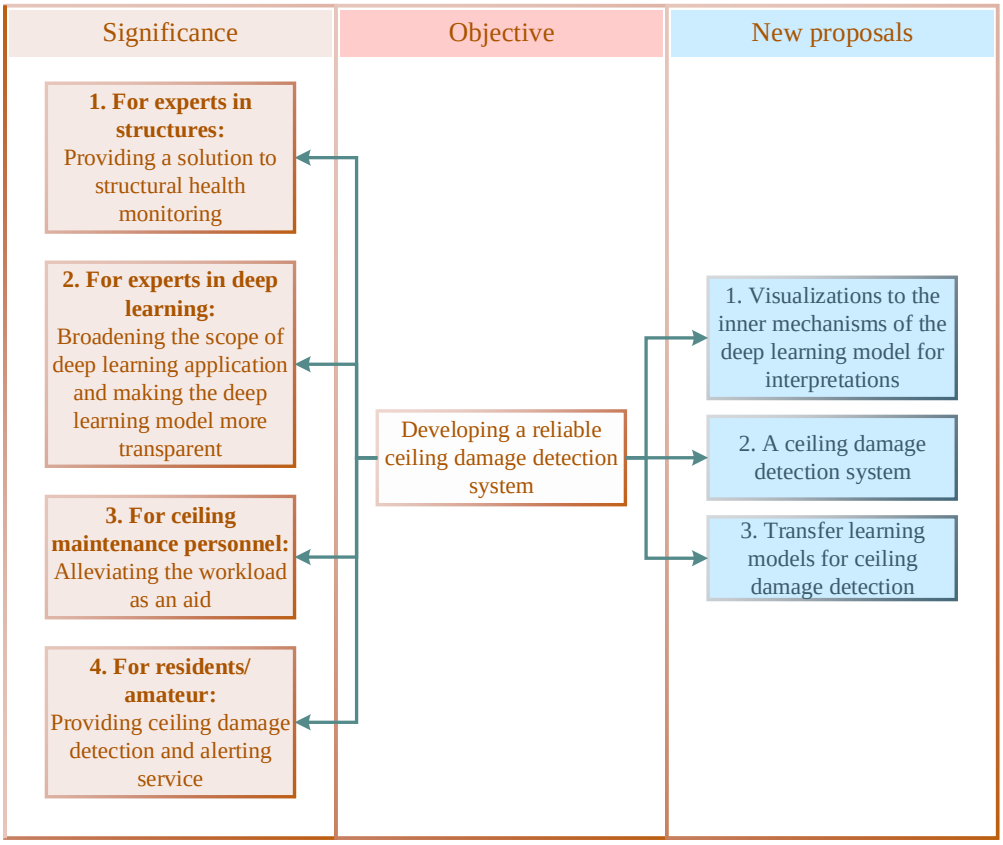


Fig. 1.15 Significance, objective and new proposals

In this thesis, the new proposals are:

(1) Visualizations to the inner mechanisms of the deep learning model for interpretations:

The deep learning method yields amazing results in accuracy. However, the mechanisms in deep learning are still in research to investigate the intelligence-like performances of the deep learning. In this research, the visualizations of the deep learning model provide interpretations of the model to understand how the model works, which is applied in structural engineering for the first time. Furthermore, these visualizations provide ceiling damage detection function, which is also used for the first time in structural engineering.

This part is the core of the thesis. The following visualizations are performed:

a. Intermediate convolutional layer output visualization; b. Activation map: Generating images that most activate the learning units (filters in the convolutional layers or classification nodes); c. Saliency map and Grad-CAM: Highlighting the most contributing pixels in the input image to the final prediction.

(2) A ceiling damage detection system:

Based on the CNN model and the visualizations to it, a ceiling damage detection system is raised with the user involved in. The user can be a ceiling inspection specialist or a refugee under possible falling ceilings in large-span buildings. This ceiling damage detection system helps the user in the large-span building ceiling damage detection task.

(3) Transfer learning models for ceiling damage detection:

Transfer learning is used in this thesis to obtain more powerful deep learning models in ceiling damage detection for the first time in structural engineering.

1.4 Objectives of this thesis

The ultimate objective of this thesis is to develop a reliable ceiling damage detection system that anyone can use to receive aids from (an expert in ceiling maintenance or a layman / refugee who wants to know if the ceilings are safe).

The ceiling damage detection task is possible to be converted into ceiling image processing task. Nowadays it is very easy to take high resolution images even using mobile phones. However, there are obstacles for ceiling damage detection using image processing:

In the first place, there are many forms of ceilings and many construction methods in the ceiling industry. The reasons that cause damages in ceilings are various as well. The damaged forms in ceilings are numerous. It is difficult to generate reliable evaluations from only one ceiling image.

Secondly, although taking ceiling images is easy for anyone using mobile phones, it is difficult to judge the existence of damage region in the ceiling, especially in a large-span structure.

Thirdly, neither the user or the AI are 100% correct, opportunities to correct mistakes should be provided to both the user and the AI.

In summary, it is very difficult to come to the correct conclusions from only one ceiling image. A ceiling damage detection system with user-CNN interactive process (Interactive-AI) is devised in this thesis that can provide higher precision results by using gradually complementing process (zoom-in the original image / taking new photos). At the same time, the Interactive-AI is required that the user can grasp the

process of information analysis in the CNN model to the extent that even a layman can understand roughly whether it is a correct or an incorrect prediction. Furthermore, when the prediction is incorrect, the input image should be preserved as future training dataset for the improvement to the CNN model.

Fig. 1.16 shows the objective of this thesis:

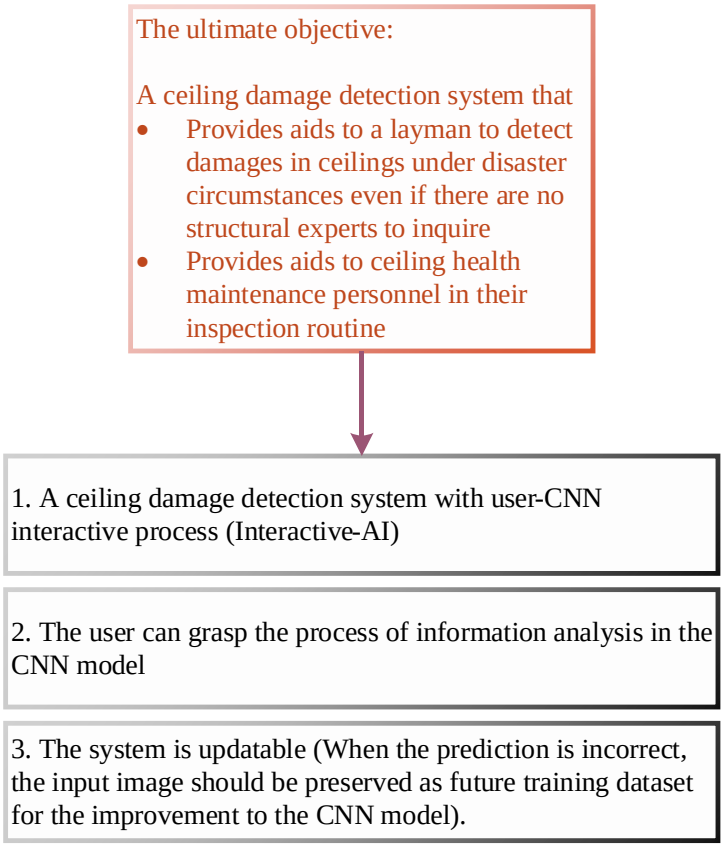


Fig. 1.16 The objective of this thesis

1.5 Overview of this thesis

Chapter 1 introduces the background of ceiling damage detection and machine learning. The requirements for the machine learning / deep learning model are raised to successfully fulfill the ceiling damage detection function before the model is really built and trained. Chapter 1 introduces the application of deep learning in structural engineering and civil engineering and the positioning of this thesis in damage detection using deep learning. The research objectives and the outline of this thesis are also proposed.

Chapter 2 explained the deep learning and convolutional neural networks in theory. The

mathematical calculations and mechanisms are introduced to build the calculation foundation to this thesis.

Chapter 3 describes the generation of a CNN model for ceiling damage evaluation. In this chapter, the datasets under the label criteria specialized for deep learning are labeled by human firstly. Then a CNN model is built and trained from scratch using countermeasures to overfit. Finally, the trained CNN model is tested and reaches a relatively high accuracy in prediction.

Chapter 4 investigates the trained CNN model to reveal the mechanisms of the prediction process. In this chapter, the trained CNN model is demonstrated from many visualization perspectives: Firstly, the outputs of the intermediate convolutional layers are visualized to grasp basic perceptive interpretations of the notions the model has learnt; Secondly, the activation maps to the filters are visualized to show what the CNN model has learnt through the gradual abstractions among the convolutional layers; Thirdly, the saliency maps and the Grad-CAM methods are used to visualize the pixels those contribute most to the final prediction to a given image. The visualization to these pixels does not only confirm that the CNN model has learnt the ‘intact’ and ‘damaged’ notions, but also provide the solutions to ceiling damage detection. Finally, a ceiling damage detection system using the blocks of CNN mode generation and visualization is raised to provide aid to both professionals of ceiling structures and common users.

Chapter 5 provides the implementations of pretrained CNN models for ceiling damage detection. Firstly, two new CNN models (transfer learning models) using the trunks of VGG16 and VGG19 are built and trained for ceiling damage detection to relatively high accuracies. Secondly, the visualization for the final two prediction nodes are visualized to confirm that the transfer learning models have learnt the most important features of intact and damaged ceilings. Thirdly, the saliency map and Grad-CAM of these two models are visualized to confirm that the transfer learning models can learn faster and better than the CNN model built from scratch. The transfer learning is an improvement to the previous model. Finally, a ceiling damage detection system using transfer learning is raised.

Chapter 6 concludes the main conclusions in this thesis and looks into future research on applying deep learning in ceiling damage detection.

2. Deep Learning and Convolutional Neural Networks

In this thesis, deep learning and convolutional neural networks are adopted to build a ceiling damage detection system. Understanding capacities and limitations of deep learning and convolutional neural networks is the foundation of building a reliable and robust deep learning model. It is also the first step for further investigating and improving the whole system [73].

2.1 Deep learning history

The expression “deep learning” is relatively new to what it really stands for. In fact, deep learning dates back to the 1940s [73], since when there have been ups and downs in the field of machine learning and deep learning. Before deep learning gets its fancy name in today, it was named “cybernetics” in the decade of 1940s for investigating how learning occurs in the brain [74, 75]. It built simple functions to reflect inputs to outputs.

The second resurgence of neural networks happened in the 1980s for ten years with the name of “connectionism” in the contest of cognitive science. Connectionism believes that a network of a large amount of simple computational units can reveal intelligence if activated [76, 77]. In this period, some fundamental concepts were established and remain significant concepts in today’s deep learning. One of them is “distributed representation” [78], which means an input can be represented by many features and each feature can be extracted from the input. Features can be expressed in many ways, abstractly or intuitively. For example, an image of red bird has features of redness, shapes combination of a bird, a beak and an eye next to the beak. There are almost countless features even in a very simple image. An artificial intelligence system can do a lot of things even if it only learns quite few abstraction and combinations of these features. The second significant concept is how to make the artificial intelligence system really “learn”. The answer is back-propagation, which transfer the error of prediction to real label back to the start of the neural networks during which process the weights in the neural networks are updated. This second resurgence of neural networks lasted to the middle of 1990s till neural network researchers could not fulfill unreasonable expectations raised by ventures and some AI technologies. Furthermore, other machine learning fields like kernel machines and graphical models made achievements in many important fields [79-82]. These two factors both declined the popularity in neural networks till the year of 2006. During this period, the Canadian Institute for Advanced Research (CIFAR) launched the Neural Computation and Adaptive Perception (NCAP) research program, which kept the neural networks

research alive and united research groups led by Geoffrey Hinton, Yoshua Bengio and Yann LeCun respectively.

In 2006, a breakthrough in the research of neural networks brought the third wave of deep learning till the time of today. Geoffrey Hinton reduced high-dimensional input into low-dimensional codes by training a multilayer neural network called a deep belief network using greedy layer-wise pretraining [83]. Soon researchers found that this strategy could be transferred to train many other kinds of neural networks [84, 85]. These researches propagated the term of "deep learning" widely to declare to both the researchers and the public that now neural networks were able to train deeper and more complicated neural networks than before. Researchers could focus on the architecture design and theoretical exploration of neural networks [86-88]. In the third wave of deep learning, the deep neural networks have outperformed many other AI systems in machine learning. Looking for the combination of deep learning with elder machine learning algorithms to expand application of AI technology is pushing the border of AI much more further [89, 90]. Increasing dataset sizes play a key role in the development of deep learning because training the learning algorithms requires sufficient data which were hard to obtained before the "big data age". Learning with small size of data is also an important research area when researchers have already designed algorithms reaching human performance. Another important role is the increasing models size that accelerates the development of deep learning. Computational resources were very expensive in the 1980s, which restricted the scale of the neural networks. With cheaper and faster CPUs are available, the model size of the neural networks could be larger and more powerful in tasks. Later researchers found that GPUs were more suitable for the calculation of network and introduced GPUs into deep learning. Faster network of deep learning and more robust software infrastructures of distribution computing provide general tools for people who owns only a good graphics card if he / she is interested in deep learning. Here came the boom of deep learning research.

Object recognition using deep models back to the date of 1980s, a neural network using back-propagation learns with the weights updating [77]. For the development of more than twenty years, modern object recognition neural networks become more complex and robust, greater size and resolution images are inputs to deep learning networks [91-93]. A large contest of object recognition held each year is the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). A convolutional network solution of neural network dominated the winning position in 2012, this was also the time GPUs became shine in deep learning field [94]. Since then, the top positions of image recognition were taken by convolutional networks. Since object recognition is reaching or even outperforming human level, image segmentation became popular in deep learning. Object detection and semantic segmentation can solve so many problems that it would be a breakthrough in many fields like robotics, self-driving and almost any visual

algorithms that were thought impossible before [95-102].

Tasks that deep networks can solve become more complex with the size and accuracy of the deep networks increase. A team from google showed that neural networks was able to learn the whole sequence of numbers or characters in an image which was believed such kind of learning only can be realized by labeling digits or characters one by one [103]. Deep learning is also getting achievements in speech recognition [104]. The introduction of deep learning into speech recognition drop the error rates rapidly. Another achievement of deep learning is the combination to reinforcement learning. Reinforcement learning is characterized as an unsupervised agent exploring the environment defined by program, learning how to best fit the environment by getting reward or punishment when different event occurs. DeepMind project launched by google has astonished the world in 2017 with AlphaGo and AlphaGo Zero.

Deep learning now is almost used by all top technology companies and other worldwide leading companies. The prospect of deep learning and machine learning propels resources and attention infuse in the research and application of AI field. Progresses in deep learning also depend on the perfection of infrastructures in software. TensorFlow [105], Theano [105, 106], Torch [107], Caffe [108], Keras [109] are all using in research and software development. Deep learning also makes the GPU producer NVIDIA profit grown rapidly in recent years since GPUs are found much more faster in deep learning than CPUs.

Deep learning also inspires other sciences and makes contributions to them. Convolutional neural networks and transfer learning provide diagnose accuracy better than experts in skin cancer [110]. Deep learning also provides tools for preprocessing data, building a deep model to grasp the characters of the data and make predictions to new data. It can predict how molecules interact with each other to help companies in designing new drugs [111].

In summary, as a branch of machine learning, deep learning has been using large amounts of human knowledge of brain research and has gain fruity results in applying in many fields for the past decades. With more data, more strong calculation power and more dedicates of researchers and programmers, the future of deep learning is full of challenges and opportunities to push the border of AI further to new frontiers.

2.2 Deep neural networks

2.2.1 Perceptron

The term perceptron dates back to the year of 1958 in the brain science [112]. One perception is a mathematical function mapping some set of inputs to output value. It is used as a binary classifier in supervised learning, meaning the output to one specific perceptron is either 1 or 0 (yes or no). A perception can be expressed as:

$$f(x) = \begin{cases} 1, \text{ if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0, \text{ ohterwise} \end{cases} \quad 2.1$$

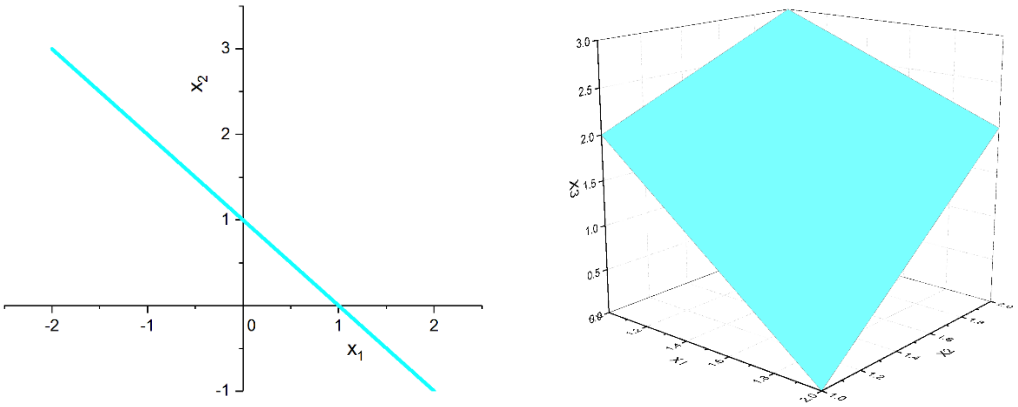
Where: w_i is weight of the perceptron, x_i is one of inputs to the perceptron (n total inputs), b is the bias.

Notice $\sum_{i=1}^{n} w_i x_i + b$ in Eq. 2.1 is an n-dimensional linear function. Fig. 2.1 shows perceptron of dimension two and three when calculating $\sum_{i=1}^{n} w_i x_i + b$. A perceptron over three dimensions is hard to visualize but shares the same measures in optimizing w and b to function as a binary classifier. Fig. 2.2 shows the outputs of a perceptron, in this process, decisions are made to identify the inputs label: 0 or 1 using Heaviside step function. Fig. 2.3 shows the whole process when a perceptron calculates inputs to output (0 or 1). During this process, a perceptron can determine the label of the inputs by fixed weights (w_i) and bias (b).

Determining proper weights (w_i) and bias (b) is the key to validate a perceptron. This is also when "learn" happens in a perceptron. Tuning the weights (w_i) and bias (b) to proper values using data is called "train". Trained weights and bias are ready to make prediction to new inputs.

The processing that a perception does is straight forward and only suitable for a quite narrow range of problems. In the real world, inputs of a specific problem are usually too complex to be interpreted by linear function and inner data structures in the inputs

are implicit to be discovered by only one perceptron. From the perspective of biological neural scientists, a perceptron can be interpreted as a mathematic model of a neuron in the real world (Fig. 2.4). By connecting perceptrons, it is mimicking the structural of a brain. The system connected by perceptrons is named neural networks.



2-dimension (x_1 and x_2) 3-dimension (x_1, x_2 and x_3)

Fig. 2.1 Perceptron of dimension of two and three

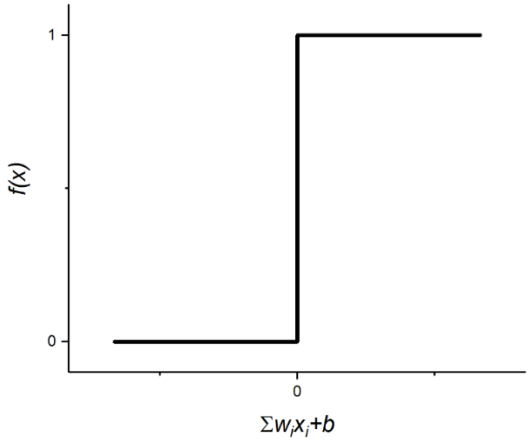


Fig. 2.2 Perceptron outputs (Heaviside step function)

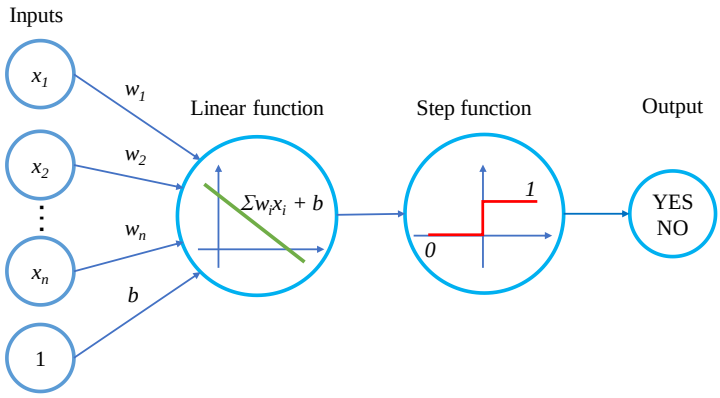
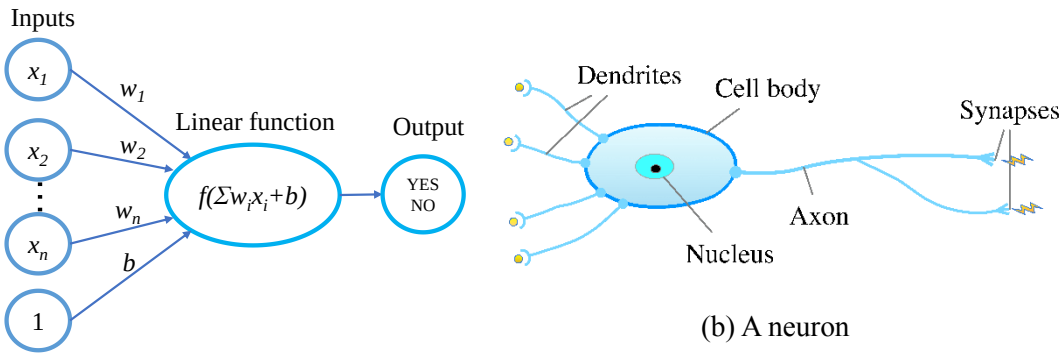


Fig. 2.3 A perception processing inputs



(a) A perceptron

(b) A neuron

Fig. 2.4 From perceptron to neural networks

Fig. 2.5 shows an example of perceptron algorithm calculating parameters (weights (w_1 and w_2) and bias (b)) of a perceptron in a two-dimension space. Suppose we have known a bunch of points with coordinates (x_1, x_2), each point is labeled as y ($y=0$ if blue, $y=1$ if orange). The task is to find a perceptron with proper parameters that best separates these two kinds of points. This is a classical classification problem. We can tell that the line best separates these points is the red line from our human instinct. A computer concludes the right answer by perceptron algorithm, which is also the process that a perceptron learns from data.

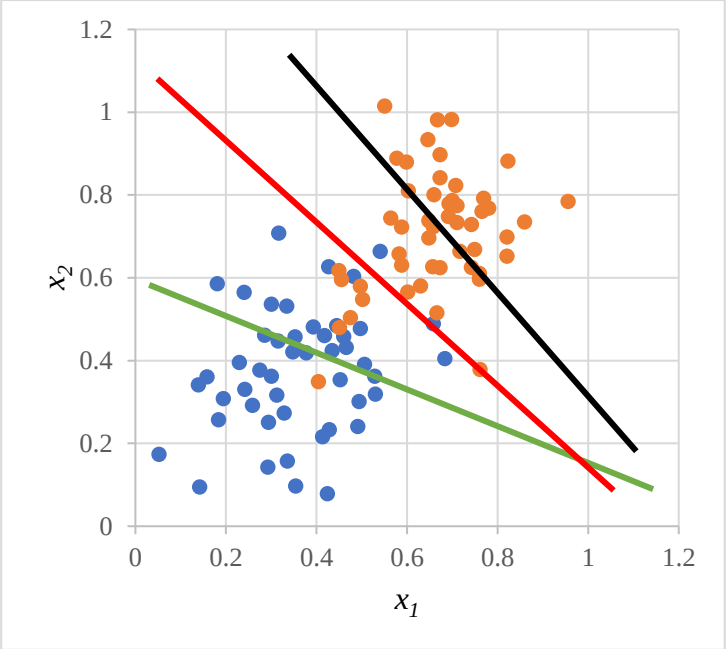


Fig. 2.5 Perceptron algorithm

The perceptron algorithm can be expressed as the pseudocode below:

1. Start with the definition of the perceptron:

   $y = f(x_1, x_2) = STEP(w_1 x_1 + w_2 x_2 + b)$, where STEP stands for Heaviside step function in Fig. 2.2; w_1, w_2 and b are randomly initialized.

2. For every point in the train dataset, (x_{train1}, x_{train2}) and y_{train} are already known ($y_{train}=0$ if blue, $y_{train}=1$ if orange), calculate the label to the point:

   $$y_{train} = f(x_{train1}, x_{train2}) = STEP(w_1 x_{train1} + w_2 x_{train2} + b),$$

   (y_{train} is the prediction to the train point by the perceptron);

   if $y_{train} = y_{train}$, the point is correctly labeled, do nothing;

   else, the point misclassified:

3. For every misclassified point i (n_1 … n_n):

   3.1. If $y_{i\_train} = 0$: $w_1 \leftarrow w_1 + \alpha x_{i1}$, $w_2 \leftarrow w_2 + \alpha x_{i2}$, $b \leftarrow b + \alpha$;

3.2. If $y_{i\_train} = 1$: $w_1 \leftarrow w_1 - \alpha x_{i1}$, $w_2 \leftarrow w_2 - \alpha x_{i2}$, $b \leftarrow b - \alpha$.

Where: α is the learning rate, meaning how fast the perceptron learns.

The pseudocode above shows the process a perceptron learns. It uses the idea of gradient descent to the final perceptron parameters. It also implicitly requires enough points for learning are required, which is also a crucial premise in deep learning.

The function of Eq. 2.1 is an activation function in deep learning. Activation functions have different properties in processing digits transferred into the nodes and different backpropagation derivatives for different problems.

2.2.2 Multilayer perceptrons

One perceptron is only capable of labeling inputs as 0 or 1, which is too simple to solve real world problems. Fig. 2.5 shows a series of points easily separated by a straight line. What if the points are like that shown in Fig. 2.6? They are not separated by only one line. Now it is possible to combine two perceptrons represented with two lines (the red and green lines) to separate these points as show in Fig. 2.6. Points both to the right of the red line and to the upper part of the green line are labeled as orange, otherwise are labeled as blue. In fact, this is a logical operator "AND", which can be represented by Table 2.1. The architecture of these two perceptrons are shown in Fig. 2.7. From this perspective, other logical operators such as "OR", "NOT", "XOR" are all possible to be represented by multilayer perceptrons, which is also named neural networks. Layers between the input layer and output layer are called hidden layers. There could be as many as possible hidden layer in a neural network to output the final results.

Now it is possible to build more complex multilayer neural networks with perceprons. When there are more than two classes to divide in a series of data, the architecture shown in Fig. 2.8 is a general solution. The S-shape perceptron is a perceptron using the perceptron function of Sigmoid. A sigmoid function is:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

The shape of sigmoid function is shown in Fig. 2.9. There are two reasons sigmoid function is used wider than Heaviside step function in neural networks: (1) sigmoid function is more easily calculated in differential calculus, which is important in the backpropagation. (2) the output of a sigmoid function is between 0 and 1, which can be

interpreted as a probability.

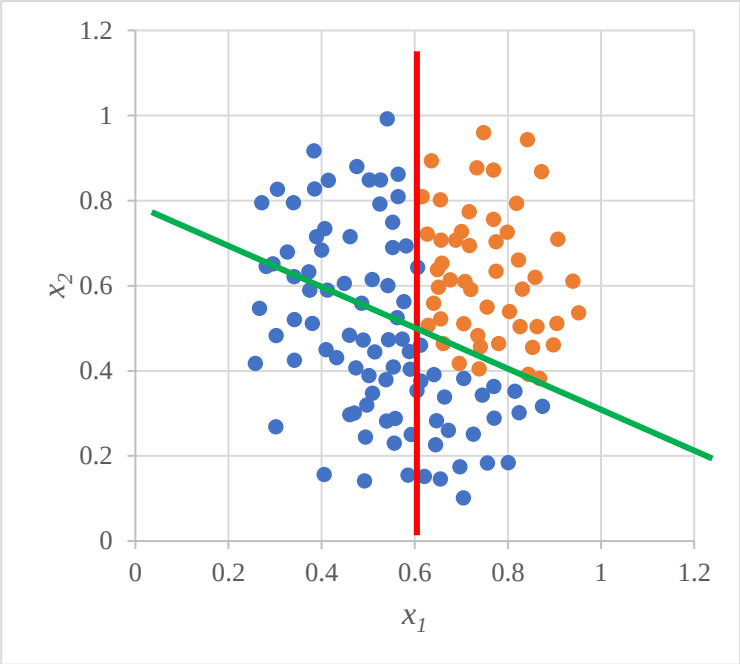


Fig. 2.6 Points unable separated by only one straight line

Table 2.1 “AND” logical operator represented by two perceptrons		
Perceptron RED line label	Perceptron GREEN line label	Final label
1	1	1
1	0	0
0	1	0
0	0	0

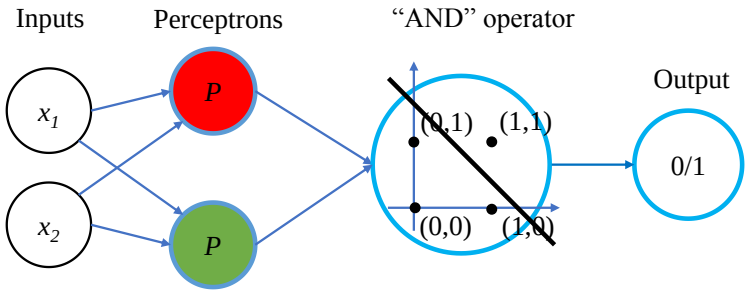


Fig. 2.7 Architecture of “AND” logical operator by two perceptrons

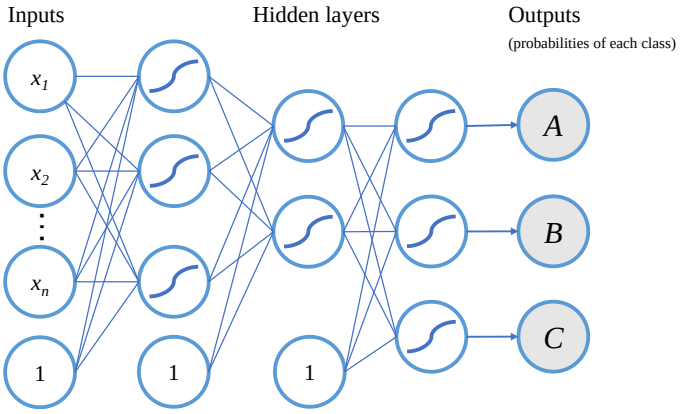


Fig. 2.8 Multi classes classification

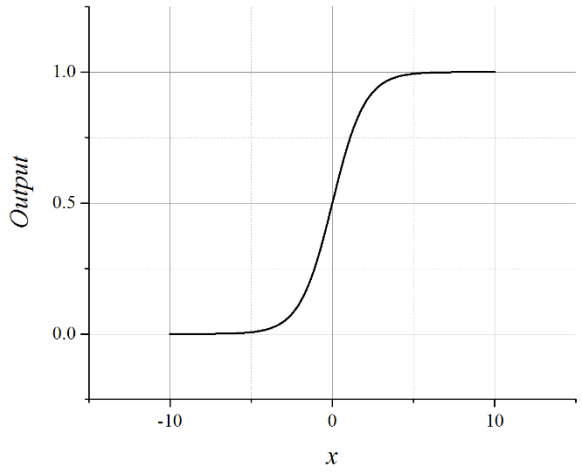


Fig. 2.9 Sigmoid function

A simplest neural network can be expressed as inputs passed out to an activation function, *f(h)*, which can be step function or sigmoid function:

$$y = f(h) = f(\Sigma w_i x_i + b) \qquad 2.3$$

By adding layers and nodes to a neural network, it becomes powerful in processing and analyzing data.

2.2.3 Feedforward, error function and backpropagation

Feedforward is the process that a neural network turns inputs into an output. The last section has introduced the basic intuition of how the data flows from input to output. The mathematical calculations of a neural network shown in Fig. 2.10 can be expressed as:

$$y = \sigma\left[\begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix} \bullet \sigma\left(\begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \bullet \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}\right)\right], \ \sigma() \text{ is the sigmoid function} \qquad 2.4$$

Sigmoid function in this neural network is called “activation function”. There are many other kinds of activation functions in the neural network.

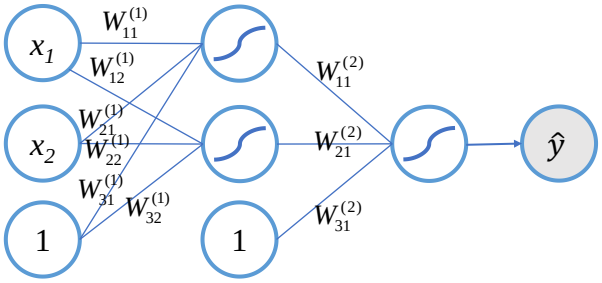


Fig. 2.10 Feedforward of a neural network

Error function is to evaluate the difference between prediction and the labeled value. Error function is also named loss function. The learning process is also the process that minimizing the error function by adjusting weight parameters W. The error function to the perceptron in Fig. 2.10 can be expressed as (cross-entropy formula):

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(y_i) + (1-y_i)\ln(1-y_i) \right] \qquad 2.5$$

The error function in Eq. 2.5 suits for only problems of two classes (like labeling class A = 1, class B = 0). When there are multi-classes in one neural network (for example, Fig. 2.8 has three classes), the error function is (cross-entropy formula):

$$E(W) = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \ln(y_{ij}) \qquad 2.6$$

Backpropagation is the process that really makes the whole neural network learn [90,

113]. It uses the idea of gradient descent. In general, backpropagation consists of:

1. Pass the input x through the neural network to outputs y by feedforward operation.

2. Compare the outputs y from step 1 with the input label y, which is the desired output.

3. Calculate the error $E(W, x, y)$, which can be represented in many kinds of error functions.

4. Run another feedforward operation from backwards (backpropagation) to spread the error to the weights (including the biases). Thus, update the weights to get a better model.

5. Repeat step 1 to step 4 using more labeled data until the model is good enough.

The purpose of the backpropagation is to figure out the partial derivatives of the error function respect to each individual weight in the neural network. In math, backpropagation to the neural network in Fig. 2.10 and Eq. 2.4 and be expressed as:

$$y = \sigma W^{(2)} \circ \sigma \circ W^{(1)}(x) \tag{2.7}$$

Eq. 2.7 is equivalent to Eq. 2.4, where:

$$W^{(1)} = \begin{pmatrix} W_{11}^{(1)} W_{12}^{(1)} \\ W_{21}^{(1)} W_{22}^{(1)} \\ W_{31}^{(1)} W_{32}^{(1)} \end{pmatrix}, \; W^{(2)} = \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix}, \tag{2.8}$$

weight matrix of the whole neural network: $W = \begin{pmatrix} W_{11}^{(1)} W_{12}^{(1)} W_{11}^{(2)} \\ W_{21}^{(1)} W_{22}^{(1)} W_{21}^{(2)} \\ W_{31}^{(1)} W_{32}^{(1)} W_{31}^{(2)} \end{pmatrix}$

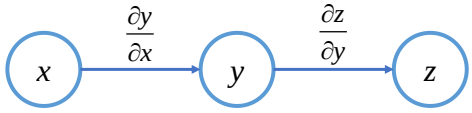
Error function:

$$E(W) = E(W_{11}^{(1)}, W_{12}^{(1)}, \ldots, W_{31}^{(2)}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(y_i) + (1-y_i)\ln(1-y_i) \right] \tag{2.9}$$

$$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial W_{11}^{(1)}} & \frac{\partial E}{\partial W_{12}^{(1)}} & \frac{\partial E}{\partial W_{11}^{(2)}} \\ \frac{\partial E}{\partial W_{21}^{(1)}} & \frac{\partial E}{\partial W_{22}^{(1)}} & \frac{\partial E}{\partial W_{21}^{(2)}} \\ \frac{\partial E}{\partial W_{31}^{(1)}} & \frac{\partial E}{\partial W_{32}^{(1)}} & \frac{\partial E}{\partial W_{31}^{(2)}} \end{pmatrix} \tag{2.10}$$

Update the whole weight matrix by adding $\alpha \bullet \nabla E$, α is the learning rate.

Recall the chain rule in calculus:



$$\begin{aligned} \Delta z &= \frac{\partial z}{\partial y}\Delta y \\ \Delta y &= \frac{\partial y}{\partial x}\Delta x \\ \Delta z &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x \\ \frac{\partial z}{\partial x} &= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \end{aligned} \tag{2.11}$$

Eq. 2.11 shows how a small change Δx causes change in y. Eq. 2.10 can be transferred as Eq. 2.12:

$$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial W_{11}^{(1)}} & \frac{\partial E}{\partial W_{12}^{(1)}} & \frac{\partial E}{\partial W_{11}^{(2)}} \\ \frac{\partial E}{\partial W_{21}^{(1)}} & \frac{\partial E}{\partial W_{22}^{(1)}} & \frac{\partial E}{\partial W_{21}^{(2)}} \\ \frac{\partial E}{\partial W_{31}^{(1)}} & \frac{\partial E}{\partial W_{32}^{(1)}} & \frac{\partial E}{\partial W_{31}^{(2)}} \end{pmatrix} = \begin{pmatrix} \frac{\partial E}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial h_1}\frac{\partial h_1}{\partial W_{11}^{(1)}}, \frac{\partial E}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial h_2}\frac{\partial h_2}{\partial W_{12}^{(1)}}, \frac{\partial E}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial \widehat{h}}\frac{\partial h}{\partial W_{11}^{(2)}} \\ \frac{\partial E}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial h_1}\frac{\partial h_1}{\partial W_{21}^{(1)}}, \frac{\partial E}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial \widehat{h}}\frac{\partial \widehat{h}}{\partial h_2}\frac{\partial h_2}{\partial W_{2}^{(1)}}, \frac{\partial E}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial \widehat{h}}\frac{\partial h}{\partial W_{21}^{(2)}} \\ \frac{\partial E}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial h_1}\frac{\partial h_1}{\partial W_{31}^{(1)}}, \frac{\partial E}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial \widehat{h}}\frac{\partial \widehat{h}}{\partial h_2}\frac{\partial h_2}{\partial W_{32}^{(1)}}, \frac{\partial E}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial \widehat{h}}\frac{\partial h}{\partial W_{31}^{(2)}} \end{pmatrix} \tag{2.12}$$

Then update the whole weight matrix:

$$W \leftarrow W + \alpha \bullet \nabla E \qquad 2.13$$

The whole process (feedforward and backpropagation) of neural network in Fig. 2.10 is:

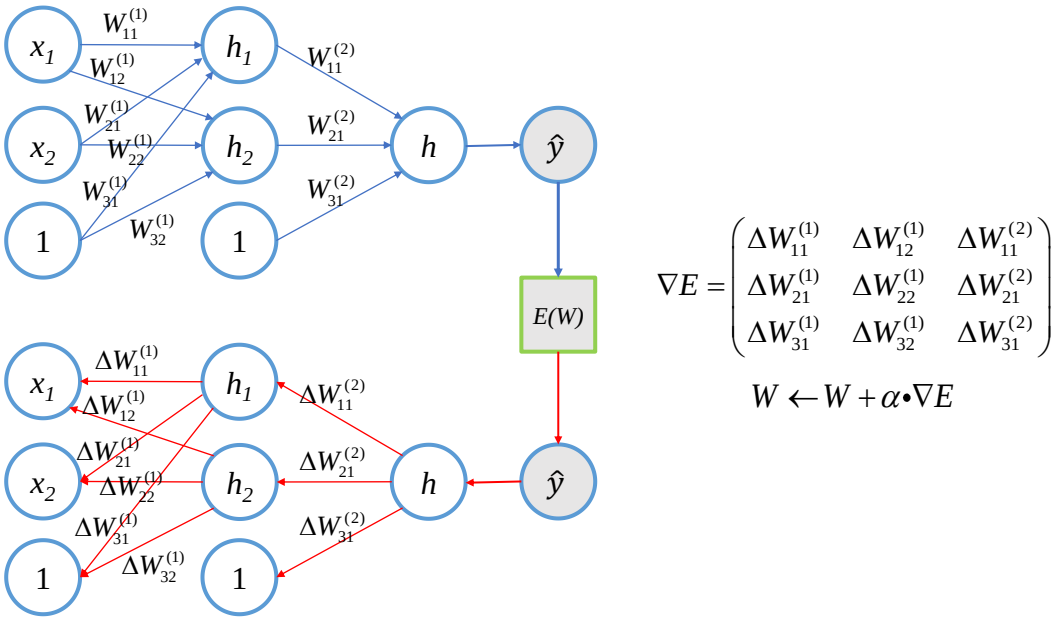


Fig. 2.11 Feedforward and backpropagation

By feeding with the neural networks enough training data and running the feedforward and backpropagation, the whole neural network learns from the training data till an acceptable prediction accuracy to new input data.

A more general demonstration of neural networks with more layers, nodes and outputs is shown in Fig. 2.12. It has two hidden layers and two output nodes. Adding or removing layers and nodes, alternating functions of nodes are all adjustments to the architecture of the neural network. With the increase of data and computing power of computers, making the neural network is possible and practical. Deep learning means the neural network has many layers with different functions to fulfill the requirements of the analysis.

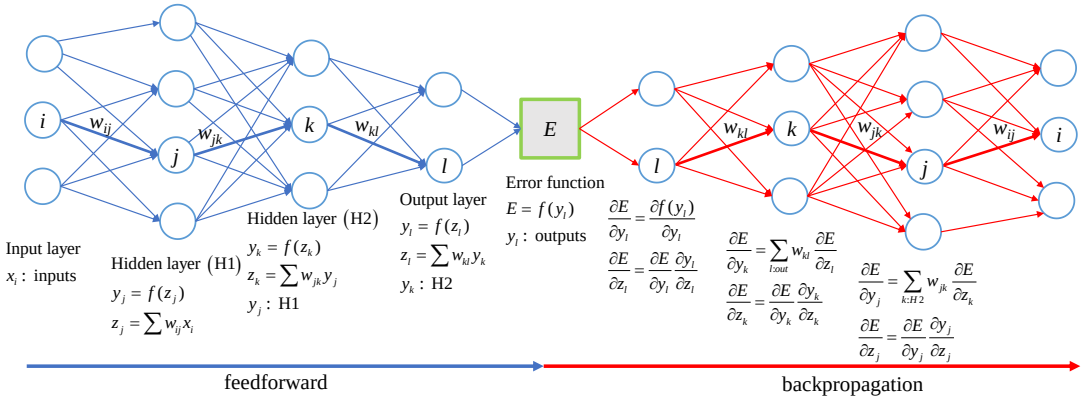


Fig. 2.12 Multilayer neural network feedforward and backpropagation

2.3 Convolutional neural networks

2.3.1 Shared weights

In mathematics, convolution refers to an operation that convolves two functions into a third one, which contains both properties of the former two functions. For example, a sound wave containing both music and human voice can be convolved with another wave that is developed to extract only human voice. The output of this convolution is only human voice. Convolution has applications in many fields that include mathematics (probability, statistics and differential equations), computer vision (image and signal processing) and natural language processing.

Convolutional neural networks (CNNs) have reached a series of state-of-art results in a variety fields including voice user interfaces, natural language processing (NLP) and computer vision. Google recently released WaveNet model using CNNs [114, 115]. The WaveNet takes in a piece of text as input and outputs an audio with a human voice reading it given other pieces of audio this human has ever read. In another way, WaveNet mimics a person's voice to an extremely similar extent. In deep learning, convolutional neural networks are especially good at recognizing objects in images and have reached state-of-art results [58, 73, 90, 116-118]. Convolutional neural networks are also called ConvNets, which are good at processing data in the form of multiple arrays. For example, 1-demensional data is signals, 2-demensional data is a grayscale image, and 3-demensinal data is a color image of RGB channels.

The main idea of convolutional neural networks is shared weights, which process inputs

variate across space. In object recognition tasks, the same class of objects usually appear in different colors, illuminations, distortions and positions in images (shown in Fig. 2.13). Objects of the same class do not change on average across time, space or other capacities but do share the same label. Building algorithms to identify classes of objects that may appear in any possible varieties is too costly or even impossible. One solution to these varieties in the same class of objects is to use shared weights. Shared weights can extract the same kind of information in different expressions if inputs' labels are the same.

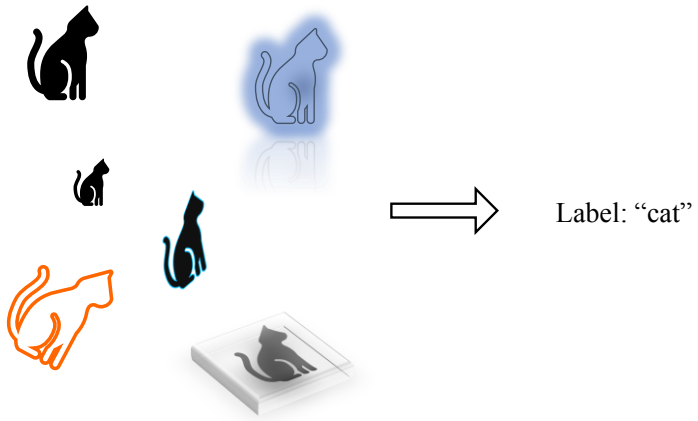
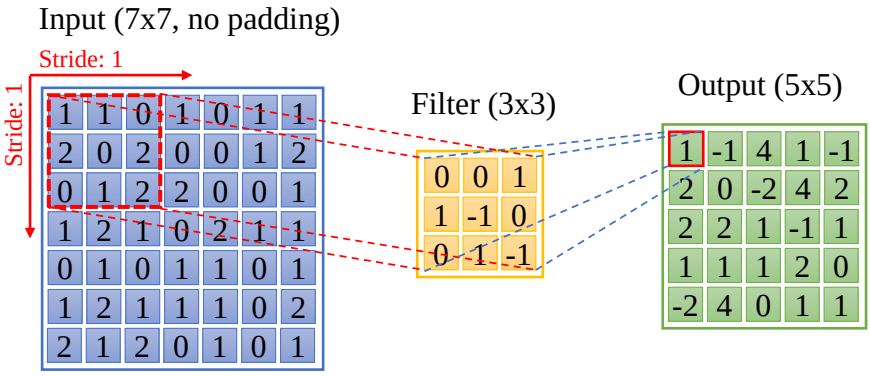


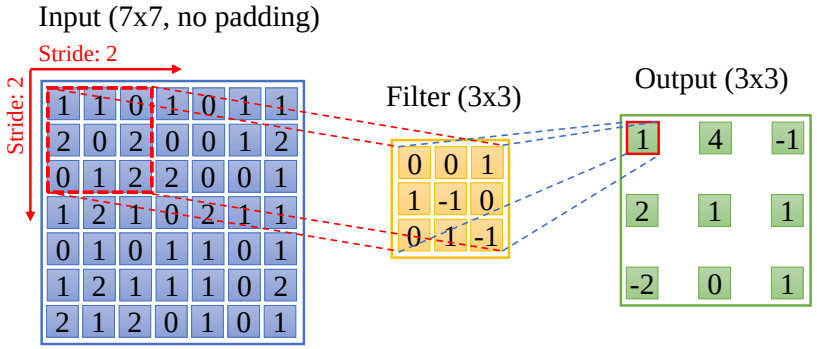
Fig. 2.13 Translation invariance and statistical invariance

2.3.2 Filters, stride and padding

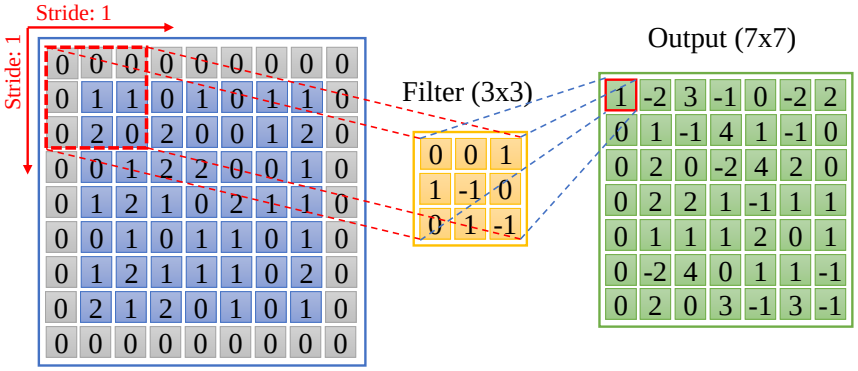
As shown in Fig. 1.6 and Fig. 1.7, an image is composed by one or multiple matrices (gray channel or color channels). The channel shape is also the shape of the image. Each pixel in one channel is valued from 0 to 255. The matrix of one channel is a 2-demensional space, composed by columns and rows showing the objects by clusters of pixels. A filter is also a 2-demensional matrix, scanning over the matrix of the image and convolve with it [116]. The output of the filter convolved with the input is also a matrix. Examples of filter, stride and padding are shown in Fig. 2.14.



(a) Stride: 1, padding: 0



(b) Stride: 2, padding: 0



(c) Stride: 1, padding: 1

Fig. 2.14 A filter convolves with a grayscale image

More generally, the calculation of the output matrix when a filter matrix scans over an input matrix is:

Input

$$\begin{pmatrix} I_{1,1} & \cdots & I_{1,w\_I} \\ \vdots & \ddots & \vdots \\ I_{h\_I,1} & \cdots & I_{h\_I,w\_I} \end{pmatrix} \underset{\text{P: padding}\\ \text{S: stride}\\ \text{h: height}\\ \text{w: width}}{\overset{\text{CONVOLVE WITH}}{\rightleftarrows}} \begin{pmatrix} F_{1,1} & \cdots & F_{1,w\_F} \\ \vdots & \ddots & \vdots \\ F_{h\_F,1} & \cdots & F_{h\_F,w\_F} \end{pmatrix}$$

Filter

Output

$$\Rightarrow \begin{pmatrix} O_{1,1} & \cdots & O_{1,w\_O} \\ \vdots & \ddots & \vdots \\ O_{h\_O,1} & \cdots & O_{h\_O,w\_O} \end{pmatrix} \tag{2.14}$$

where:

$h\_O=(h\_I-h\_F+2P)/S+1$

$w\_O = (w\_I - w\_F + 2P) / S + 1$

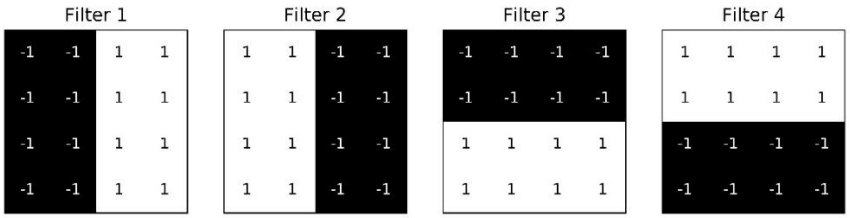
$$O_{ij} = sum\left(\begin{pmatrix} I_{i,j} & \cdots & I_{i,j+w\_F-1} \\ \vdots & \ddots & \vdots \\ I_{i+h\_F-1,j} & \cdots & I_{i+h\_F-1,j+w\_F-1} \end{pmatrix} \circ \begin{pmatrix} F_{1,1} & \cdots & F_{1,h\_F} \\ \vdots & \ddots & \vdots \\ F_{w\_F,1} & \cdots & F_{w\_F,h\_F} \end{pmatrix} \right)$$

$$= sum\begin{pmatrix} I_{i,j} * F_{1,1} & \cdots & I_{i,j+w\_F-1} * F_{1,h\_F} \\ \vdots & \ddots & \vdots \\ I_{i+h\_F-1,j} * F_{w\_F,1} & \cdots & I_{i+h\_F-1,j+w\_F-1} * F_{w\_F,h\_F} \end{pmatrix} \tag{2.15}$$

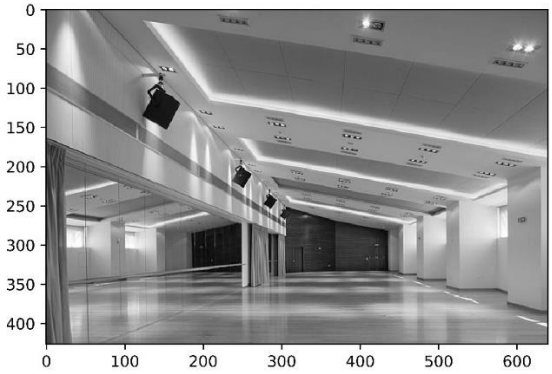
Eq. 2.14 and Eq. 2.15 show the convolution of the filter and the input. The output to the input is also called the feature map or the activation map of the input. The shape of the output is determined by the input shape, the filter shape, the stride and the padding. The stride is how fast the filter scans over the input. The padding is to provide extra input columns and rows to the input to prevent from omitting information in the input. Choosing reasonable shape of filter, length of stride and padding helps with the extraction of required information in the image.

Why are the filters adopted in the convolutional neural networks? Because the filters and how they scan over the input are the shared weights. These shared weights scan all over the input and output feature map due to the same criteria. The criteria are the properties of the filters, who focus on the locally connected subsets of the input in the same shape to the filters. Fig. 2.15 shows that different filters have different characteristic focus properties. From the features maps to the input, perceptually it can be noticed that filter 1 and filter 2 focus on vertical edges and filter 3 and filter 4 focus on horizontal edges. Moreover, filter 1 focuses on the right part edges of the image

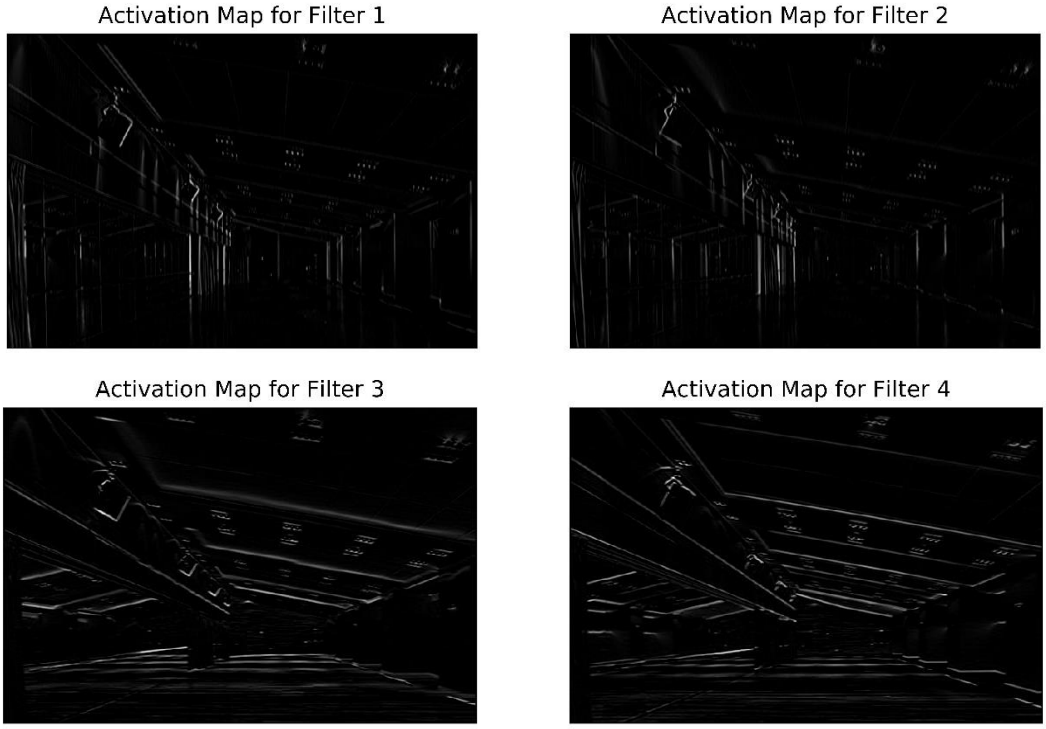
while filter 2 focuses on the left part edges. While filter 3 and filter 4 focus on the underneath and upper edges in the image. It is also possible to build filters that focus on diagonal edges or circular edges. However, it is too complex to designate specific filters to reach the purpose of object classification. Introducing the filters into neural networks is a good idea to optimize the filters by themselves.



(a) Four 4x4 filters



(b) Ceiling image (grayscale)



(c) Feature maps by filters

Fig. 2.15 Convolutional outputs by different filters

A filter convolving with the grayscale image in 2-dimensional operations is shown in Fig. 2.15. When the input image is 3-dimensional, the filters are also required in the form of 3-dimension. The convolutional operations and outputs are show in Fig. 2.16. The sub-filters combining into one filter are also shared weights over the inputs.

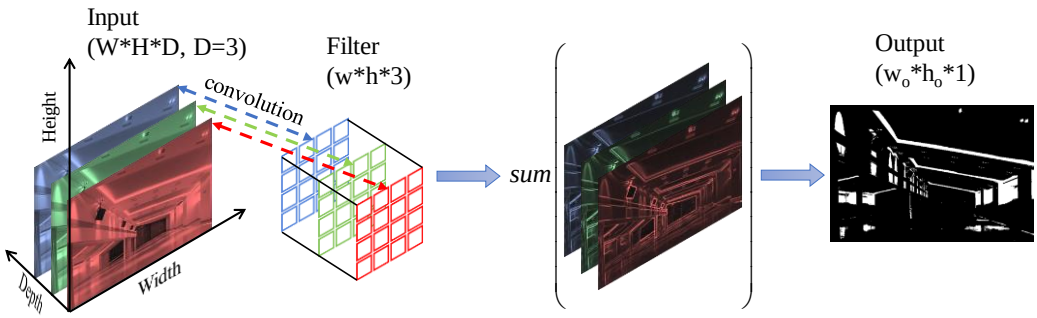


Fig. 2.16 A 3-dimensional filter convolves with a color image

2.3.3 Convolutional layers

By introducing the convolutional operation into deep neural networks, building a generally convolutional function layer with trainable parameters is the basic block, namely the convolutional layer. In image processing, the input is an image with three color channels of RGB. The filters are shared weights to the inputs. A convolutional layer consists the number of filters, who have the same shape and strides. Different filters have different biases. Fig. 2.17 shows the details of feedforward from input to the feature maps through a convolutional layer.

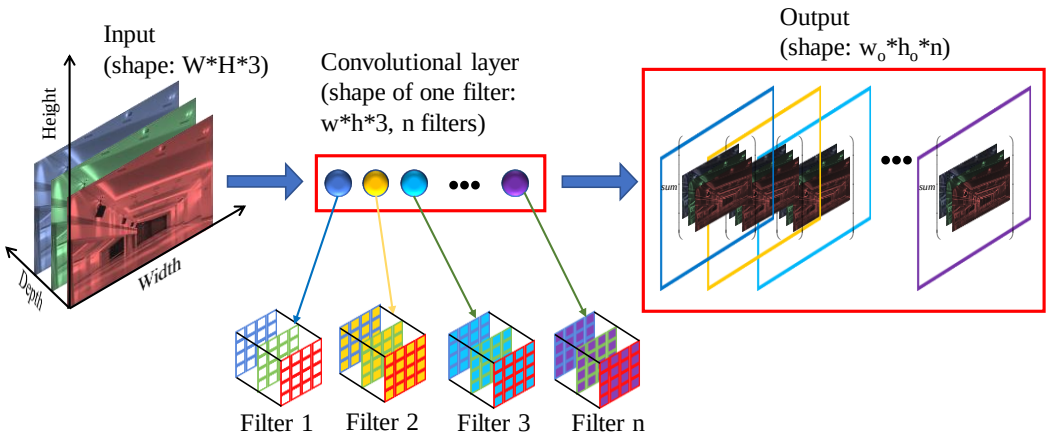


Fig. 2.17 Feedforward in one convolutional layer

Shape of input to the convolutional layer is W*H*3. The convolutional layer has n filters in it. Each filter has the same number of sub-filters to the input depth. The feature maps have the shape of w_o*h_o*n. Compare the shape of the input and the output, they

are all 3-dimensional matrices. The output of one convolutional layer can be another input to the next convolutional layer. Recall the deep neural network in Fig. 2.12, replacing the hidden layers with convolutional layers gets a convolutional neural network. A simple convolutional neural network is shown in Fig. 2.18. The input (shape as an image) is feedforwarded to three convolutional layers, objects in the image is recognized in multi abstraction. The results generated from the convolutional layers are then fed into a few fully connected layers to execute dimensionality reduction from 3-dimension to lower dimensions. Finally, the results processed by the fully connected layers are fed into the classifier to generate predictions to the input. The predictions are compared with the label to the image to form error functions for the following backpropagation. Notice the weights are the filters in the convolutional layers, randomly initialized when the convolutional neural network was built. These weights are updated through backpropagations that are the same to that in the deep neural networks.

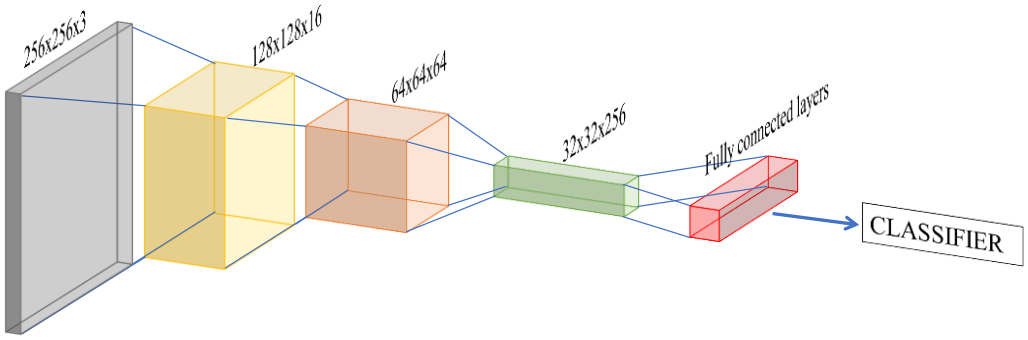


Fig. 2.18 The architecture of a simple convolutional neural network

2.3.4 Understanding the abstractions in ConvNets

Understanding how the convolutional layers do abstraction to the input is crucial in building and adjusting the whole convolutional neural networks. It is still hard to prove the process of optimization of ConvNets in mathematics. But there are many attempts to investigate what really happen in the training process and trained deep convolutional neural networks [119, 120]. The results show that the ConvNets do abstractions gradually in the convolutional layers. As is shown in Fig. 2.19, the ConvNets recognize simple shapes of objects in the first few layers, then these shapes are connected into small objects, which are parts of bigger objects. The final output is the combination of the important small object recognized by the convolutional layers.

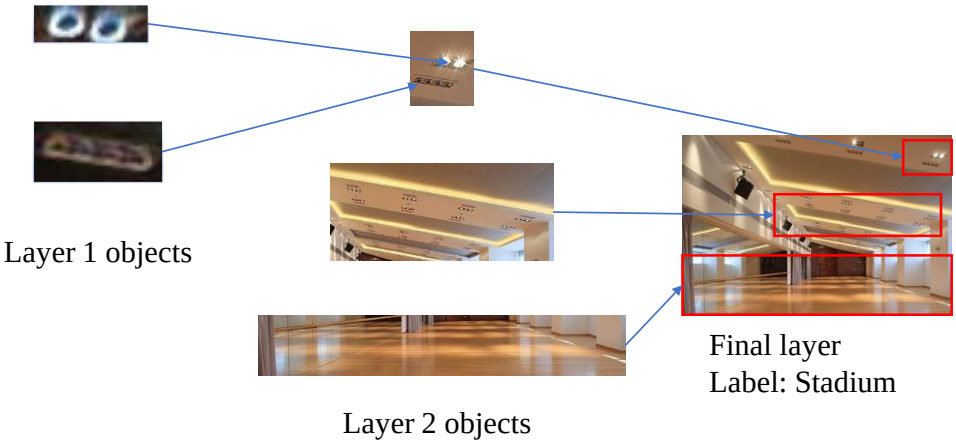


Fig. 2.19 An example of how convolutional layers recognize an image of a stadium

2.4 Conclusion

In this chapter, a brief history and the basic blocks of deep learning are introduced and explained. A deep learning model composed by many hidden layers with different functions can perform multilayer abstractions to the input, gradually reach the most characteristic feature of the input. A deep learning model is trained through many feedforward and backpropagation cycles. An important functional layer, the convolutional layer is introduced in this chapter. Convolutional layers composed by multiple filters scanning over the input to perform spatial convolutional operations are especially suitable for processing images. With the knowledge of deep learning and the convolutional layers, it is possible to build and train a CNN model for image processing task.

3. Building and Training a Convolutional Neural Networks Model

3.1 Generating a database of ceilings images using ceilings damage evaluation criteria

3.1.1 Database of ceiling images

For machine learning and deep learning, enough data are required to make the deep learning model really “learn”: to optimize parameters in all the layers. The data consists of input data and labels to each of the input data unit. The input data is a series of data flow with space and time sequences which can be processed by deep neural networks. The labels of the input data are labeled by human. The labeling process is when human injects their intelligence into the data. Different people would label different labels to the same data. The deep neural networks would learn these labeled data to mimic the criteria that the person who labels the data obeys.

In this thesis, labeling ceiling images with commensurate dangerous degree is crucial for the following deep learning model to learn and optimize. The Kawaguchi Lab crew has been investigating and analyzing the ceiling fall accidents in Japan since the Great Hanshin-Awaji Earthquake in 1995. Thousands of ceiling images (intact and damaged) have been collected for research since then. These ceiling images are of indoor stadiums, indoor pools, bath centers, lecture halls, factories and other structures with large span. The collection of well-functional ceilings images is relatively easy, while the collection of damaged ceilings images needs efforts. Images of intact ceilings in different structural spaces are shown in Fig. 3.1. And damaged ceilings are shown in Fig. 3.2.

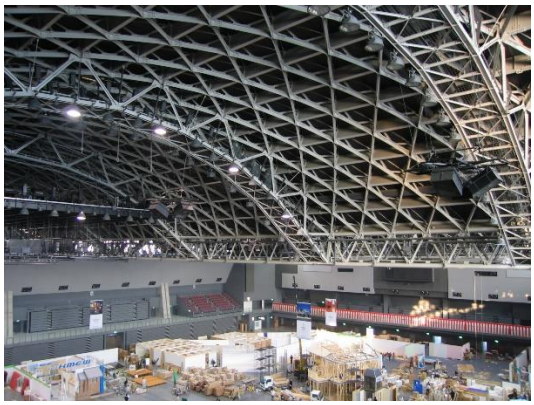
What is needed to point out is that the images in the collection of the damaged ceilings are usually severely damaged or destroyed. Because ceilings that contain very few tiny damages are hard to detect. Tiny damages are usually neglected by people until they develop into great danger to human body. The imperfection of the data is very common in real-world problems in machine learning. However, in the recognition of intact and damaged ceilings, the deep neural networks can grasp the notions of good and bad for classification.

(a) a stadium

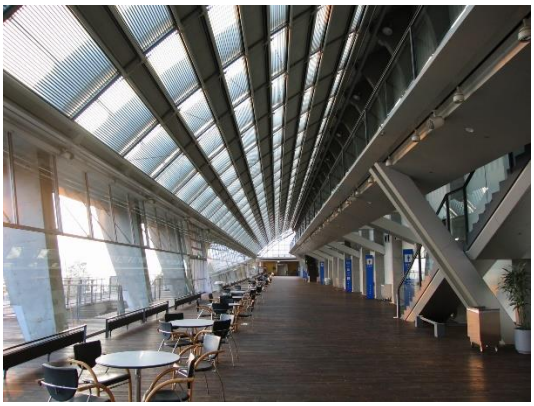
(b) an indoor swimming pool

(c) an indoor basketball hall

(d) a lecture hall



(e) an exhibition center



(f) a corridor

Fig. 3.1 Intact ceilings in different structual spaces

(a) an indoor swimming pool

(b) an indoor stadium

(c) an office

(d) a theater

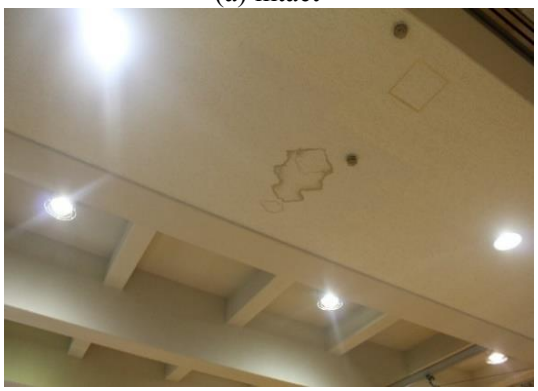
Fig. 3.2 Damaged ceilings in different structures

3.1.2 Labeling criteria

Building a reasonable labeled database for the deep learning model is the foundation of the whole research. Labeling the images is to apply the ceilings damage evaluation criteria into the images. The ceilings damage evaluation criteria are combination of ceilings knowledge and deep learning knowledge. The criteria are required to be suitable for both damage evaluation by human and learning process by deep neural networks. For simplicity, the damaged ceilings and the intact ones are basically two types of ceilings labels, which can be labeled as 1 and 0. However, the differences between damaged and intact ceilings are not as significant as that between a dog and a cat. Intact ceilings would become damaged ones due to deterioration gradually and earthquakes suddenly. Between the labels of 0 (intact ceilings) and 1 (very dangerous ceilings), there are other damage degrees that gradually increase (shown in Fig. 3.3).

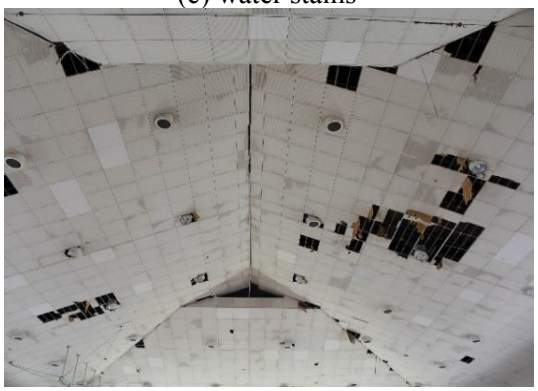
(a) intact

(b) exfoliation

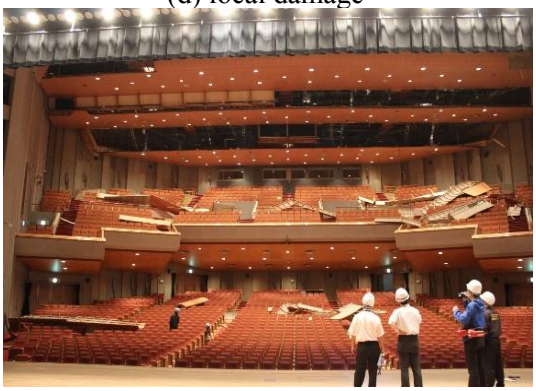


(c) water stains

(d) local damage



(e) severe damage



(f) destroyed

Fig. 3.3 Varying degrees of damages (from intact to destroyed)

To label all the ceiling images with damage degree, there is a tendency to label them as precise as possible. It is easy to label danger degree of Fig. 3.3(a) as 0 and Fig. 3.3(f) as 1. However, labeling other images in Fig. 3.3 to an exact danger degree is relatively hard. Considering the practical applications and objectives of this research, precaution to possible accidents and detecting possible damaged parts of ceilings are the highest priorities to prevent accident. The number of classification labels for the ceiling images is set as two, the label of intact ceilings (0) and the label of damaged ceilings (1), for three reasons: 1. The final objective of the research is to keep the safety of human lives and properties from damaged ceilings, even tiny damages that are possible to cause huge loss are defined ‘dangerous’, labeled as 1; 2. The number of ceiling image in the database, which is approximately 2,000 images, does not support the training for too

many labels, which will result in overfit; 3. It is difficult to quantify the extent of damages, there exist many damage forms with many different damage degrees.

The criteria for labeling are as follows:

(1) For simplicity to both human and the deep learning model, the labels are distinguished as 0 and 1, which resembling intact and damaged images respectively.

(2) Label as 0 if there are no obvious damaged parts in the image or the damaged parts are not severe and less than 5% of the whole image in proportion.

(3) Label as 1 if there are clearly damaged parts that would spread to other part in the ceilings or the whole damaged parts are more than 5% of the whole image in proportion.

Reasons for the labeling criteria are:

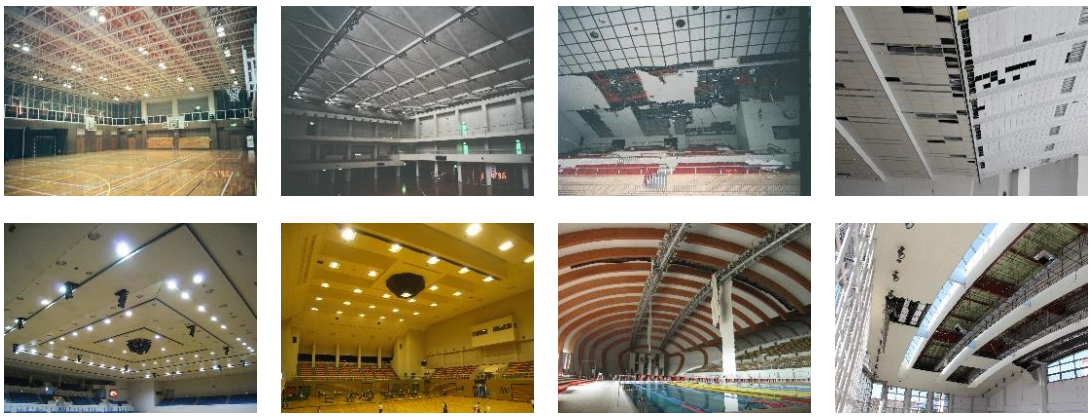
(1) The main objective of this search is to develop a ceiling damage detection system for both routine checks of the ceilings by the managers of a building and danger evaluation of the ceilings when residents take refuge under these ceilings in out bursting disasters. Forewarnings are crucial under these circumstances.

(2) Detecting ceilings damages and forewarning possible dangers from images only taken under the ceilings (the viewpoint of the observer) has its intrinsic defects: what is happening on the other side of the ceilings are intangible. However, status of the ceilings on the visible side does reflect important characters of the ceilings health, which are easy to obtain. In other words, inspecting the inner side of the ceilings is too expensive while the outside of the ceilings can provide abundant information to the safety of the ceilings.

(3) In statistical hypothesis testing, statistical significance refers to the fact that a result is unlikely to have occurred for the given null hypothesis. The significance level is typically set as 5% or lower, which means that if a fact that has very low probability (5% or lower) to occur really occurs, the null hypothesis is rejected. The null hypothesis is usually that the system is working properly. In this research, five percent is a benchmark to affirm if an incident really has occurred. In image recognition of deep neural networks, when the damaged part in an image exceeds five percent to the whole area, it is prominent enough to be noticed by the neural networks from noisy points. Thus, the ratio of five percent is chosen as the benchmark for labeling.

3.1.3 Labeled database description

Ceiling images are selected for labeling from Kawaguchi Lab ceilings image collection. The labeled database contains 1147 intact ceiling images (labeled 0) and 805 damaged ceiling images (labeled 1). In the ceiling database, the intact ceiling images are mostly taken from the global perspective that reflecting the whole status of the ceilings; the damaged ceiling images usually focus on the zoomed in damaged regions. The damage forms include: cracks in the ceiling boards and junctions, disengagement and void in the ceiling boards. Examples of these images are shown in Fig. 3.4.



(a) Intact Ceilings (1147 images labeled 0)

(b) Damaged Ceilings (805 images labeled 1)

Fig. 3.4 Ceilings Images

3.2 Building the convolutional neural networks model

The architecture of a deep neural network refers to the arrangement layouts of the multiple layers in the whole model. The arrangement is also called hyperparameters which include the function of each layer, the parameters in the layers themselves. Hyperparameters are set before the learning process of the deep learning. A proper arranged architecture of the neural network can cost less calculation and data sources and performance better.

3.2.1 Pooling layers, dense layers and fully connected layers

As is shown in Fig. 2.18, the multi convolutional layers extract information from an image (down sampling). The convolutional layers use strides to shrink the height and width of the input layers, which is an aggressive way to down sample. Using large strides in the convolutional layers causes a great loss of information. The pooling layer

[58, 116] down samples the feature maps in the locally neighborhood with only little information loss. The pooling layer outputs the combinations of locally selected characteristic nodes. There are a few pooling layers, the most common one in which is max pooling (Eq. 3.1).

$$o_{a,b} = \max \begin{pmatrix} i_{i,j} & \cdots & i_{i+k,j} \\ \vdots & \ddots & \vdots \\ i_{i+k,j} & \cdots & i_{i+k,j+k} \end{pmatrix} \qquad a,b = \frac{i-k}{s} + 1 \tag{3.1}$$

where: i for input, o for output, k for kernel (filter) size, s for stride.

An example of max pooling is shown in

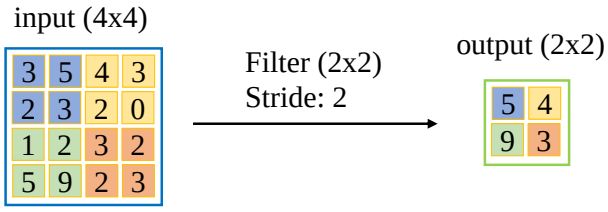


Fig. 3.5 An example of max pooing

The advantages of pooling layers are:

(1) Pooling layers do not increase any parameters.

(2) Deep neural networks with pooling layers often perform better than those without.

The disadvantages of pooling layers are:

(1) Pooing layers make the whole neural network more complex to train.

(2) More hyperparameters like the size of filter and stride of the pooling layer to choose by human.

A typical convolutional neural network[121] has the architecture shown in Fig. 3.6. The dense layer reduces a 3-dimensional matrix into a 1-dimentional matrix that neglects the inherent spatial information in the previous 3-dimensional matrix. It has no additional parameters. The fully connected layers are simply 2-dimensional matrices

with biases. They down sample the previous 1-dimentional matrix into a shorter matrix (shown in Fig. 3.7).

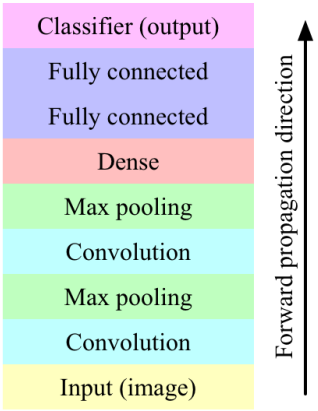
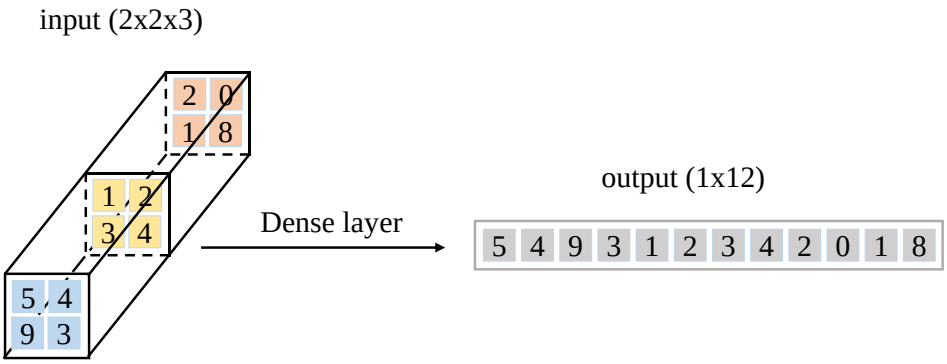


Fig. 3.6 A typical convolutional neural network architecture



(a) A dense layer

$$\overbrace{\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kn} \end{pmatrix}}^{k\times n} \cdot \overbrace{\begin{pmatrix} i_1 \\ \vdots \\ i_n \end{pmatrix}}^{n\times 1} + \overbrace{\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}}^{n\times 1} = \overbrace{\begin{pmatrix} o_1 \\ \vdots \\ o_k \end{pmatrix}}^{k\times 1}$$

where: i for input; a, b for fully connected layer parameters

(b) A fully connected layer

Fig. 3.7 A dense layer and a fully connected layer

3.2.2 The architecture of the convolutional neural networks for ceiling damage evaluation

With the blocks above, it is possible to build and train a convolutional neural network for ceiling damage evaluation. The main idea is to build functional convolutional neural networks that could be trained by the labeled ceiling images. The networks are required build in the balance of the training cost and prediction preciseness. The architecture of the CNN model is adapted from a self-driving car research by NVIDIA [122], in which the inputs are camera images of the traffic information and the outputs are steering wheel operations. The architecture of the convolutional neural networks for ceilings damage recognition follows this idea as the input is the ceiling image and the output is the digitized damage evaluation, the middle layers perform gradual abstractions through processing the input image. The architecture of the CNN model is built as shown in Fig. 3.8. This is the final architecture based on various of combinations by trial and error.

The tunable hyperparameters in the CNN model include: 1. The function determination of the layers, such as the convolutional layer, the activation and pooling layer following up with the convolutional layer; 2. The sequence of the functional layers that guarantees the information abstraction; 3. The parameters in one middle layer itself, for example in a convolutional layer, the number, the shape and stride of the filters will perform differently in training and prediction. All the items above affect the learning ability and performance of the CNN model, which is measured by the factor of accuracy in predictions. To determine the final architecture of the CNN model for ceiling damage evaluation, reasonable training process (shown in Chapter 3.3) is decisive as well. It is hard or impossible to find the most reasonable training process for a specified model.

The determination of the architecture of the CNN model mixes with the determination of the training process, in which there are many parameters and tricks to play with. The fundamental principle is: make sure that the CNN model learns as much as possible, but do not over-learn, which is also named overfitting.

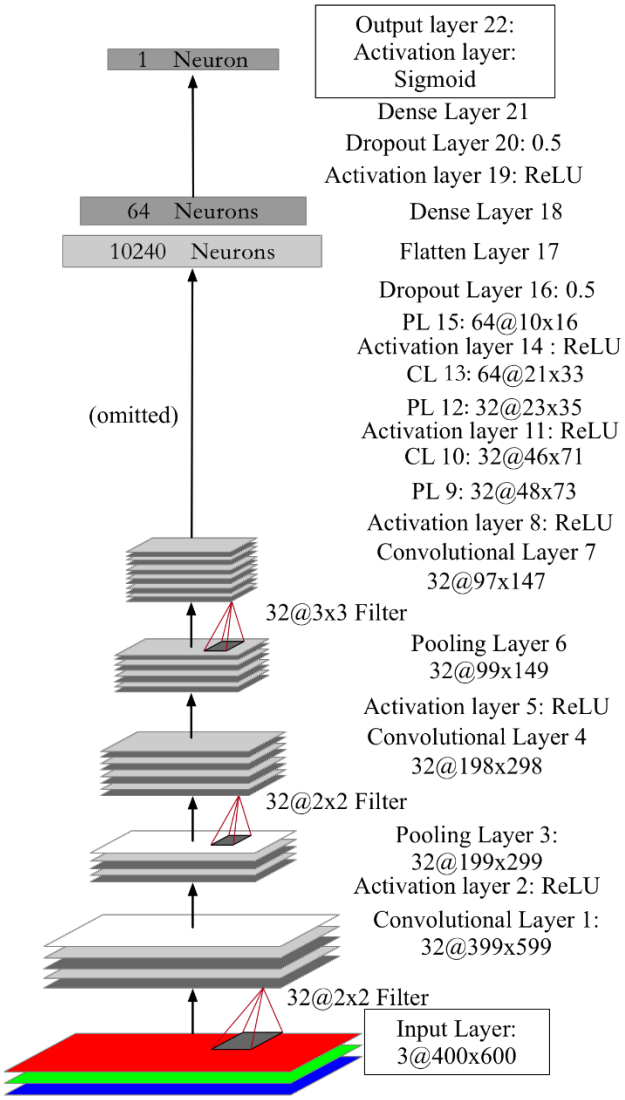


Fig. 3.8 The architecture of convolutional neural networks for ceilings damages evaluation

In the input layer, the ceilings images are resized into the resolution of 400x600x3 pixels to fit the layer. Reasons for this selection are that it can both contain enough information of the images and control the training time to some extent. The inputs are down sampled through the following layers to reach the final output of one digital node. The hyperparameters in these layers are changeable due to human judgements. Now the whole convolutional neural networks are ready to train.

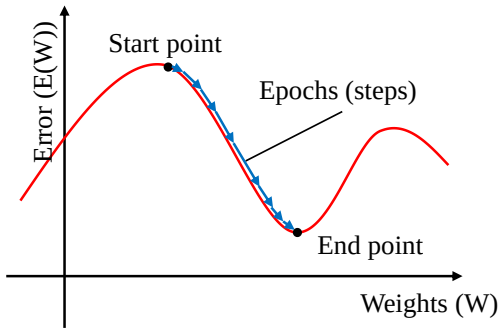
3.3 Training the convolutional neural networks model

The training process is when the learning really happens in the convolutional neural

networks. There are many parameters and solutions to choose and there are many hinders in the training process that can fail the whole neural networks. Understanding and tuning hyperparameters in the train process are crucial in deep learning.

3.3.1 Gradient descent and stochastic gradient descent

In Chapter 2, the idea of gradient descent has been introduced when updating weights in backpropagation to minimize the error function *E(W)*. The process of minimizing the error function by gradient descent can be explained in Fig. 3.9. The minimization of error happens with updates of the weight and bias in a series of steps. A step is also called an epoch that using all the data (inputs and their labels) to update weights and bias that approaching to the minimum error. The direction of updating is the negative direction of the gradient of the error function. The number of epochs is determined by confirming if the error has reached the minimum point. If it has, the minimization is terminated. In each epoch, all the inputs run through the model to calculate the error that are used to update weights and bias in backpropagation. Repeat the epochs until the error reaches the minimum.



$$E(W) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(y_i) + (1-y_i)\ln(1-y_i) \right]$$

$$y_i = \sigma(Wx^{(i)} + b)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Update W and b at each step:

$$W_i \leftarrow W_i + \frac{1}{m}\alpha(y_i - y_i)x^{(i)}$$

$$b \leftarrow b + \frac{1}{m}\alpha(y_i - y_i)$$

Fig. 3.9 Gradient descent

However, in practice one whole epoch in updating parameters of a complex deep learning model costs too much memories and calculation resources of the hardware. It is not practical to minimize the error function using the gradient descent in complex models. To solve such problem, stochastic gradient descent is invented.

Stochastic gradient descent just takes one step further than the gradient descent to successfully prevent the calculation from overflow. It requires that the data is well distributed that could be randomly equally divided into batches. A batch is a subset of the whole data. For example, a whole dataset containing one million inputs could be

randomly divided into 1000 batches that each batch only contains 1000 inputs to make each update by one batch practical on a common computer. Run the 1000 batches through the machine learning model to gradually update the weights costs much less resources. The key of stochastic gradient descent is the randomly divided batches that contain well divided patches of information in the previous whole dataset. Using these batches, the updates to the weights and bias are possible to run on the common computers.

3.3.2 Learning rate and testing

In Fig. 3.9, α is the learning rate that controls how much the model learns in each epoch. Either a too high or a too small learning rate is not economical for the minimization. If the learning rate is too high, the steps would be too big to reach the minimum and wanders around the valley bottom. If the learning rate is too low, each step is very short that would cost very long time to reach the local minimum (Fig. 3.10). Tricks of tuning the learning rate at different training stages can improve the efficiency of the training.

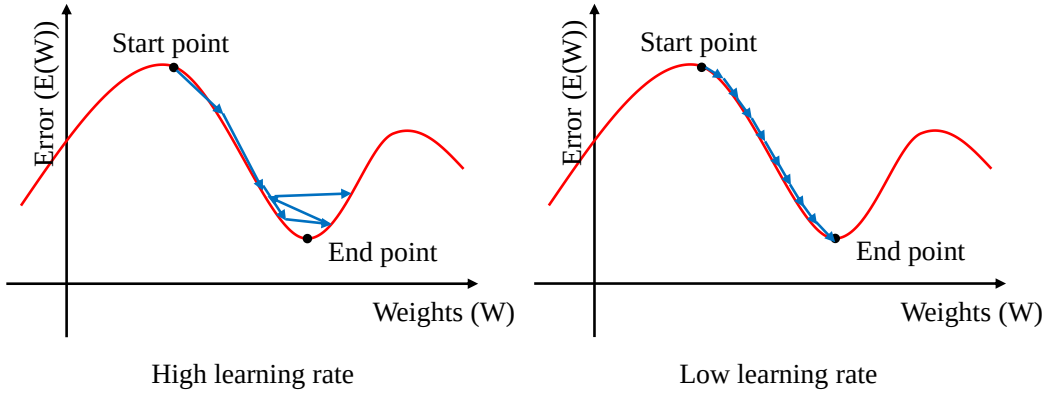


Fig. 3.10 Learning rate

Testing is the verification to the trained model to investigate if the model is good enough to make reliable predictions to new inputs. Before training the model, the labeled inputs are usually randomly divided into two sets: the training set and the testing set. The ratio of training set to testing set is usually 0.8 / 0.2, which composite the whole labeled inputs. The training set are used to update the weights in the model while the testing set is hidden from the model until the training process ends. Fig. 3.11 shows the complexities in building a model and testing it: Firstly, divide the data into training set and testing set. The models are built to classify the red and blue points. Secondly, train the models only using the training set. The trained models are the green line and the yellow curve. The green line is a relatively simpler classifying model than the yellow curve does. It can be noticed that the green line misclassifies four points in the training

set while the yellow curve classifies the points in the training set all correctly. However, when predicting to the points in the testing set, the green line has two mis-classified points while the yellow curve has three mis-classified points. The yellow curve is so complex that it remembers all the points in the training set. When training the model, there are tendencies that we want to keep the model that has the lowest error in the training set. However, a too complex model usually mechanically remembers the training data and makes unreliable predictions to new inputs. Dividing the labeled data in the first place is necessary for credible testing results.

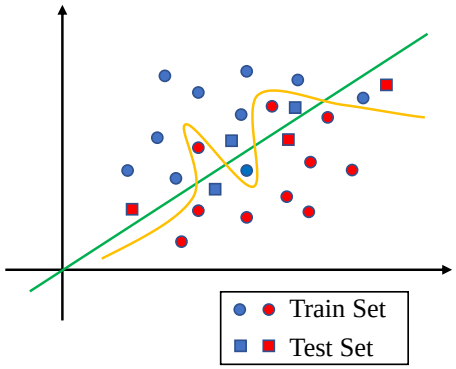


Fig. 3.11 Testing to two models of the same inputs

3.3.3 Overfitting and underfitting

It is very hard or even impossible to really find a perfect model for a problem. Usually the model is either too simple or too complex that would result in poor performances. When the model is too complex, the model remembers all the data (both the details and the noises) to learn everything reflects by the data. The learned noises and fluctuations in the data that would result poor performances to new data are overfitting. While on the contrary, when the model is too simple to really learns the data, the trained model does not even understand the inputs. Such simple models are underfitting. In Fig. 3.12, underfitting, appropriate and overfitting are shown. In deep learning, it is very hard to find the right architecture in practice. There are too many gains and losses to balance in choosing the architectures. Good news is that overfitting is sometimes acceptable if there are strains to it. The overfitting does learn the noisy and irrelevant data to the model. However, these problematic data are also from real world and do reflect the real world. Building a deep learning model that can solve complex problems are better than just doing binary linear regression.

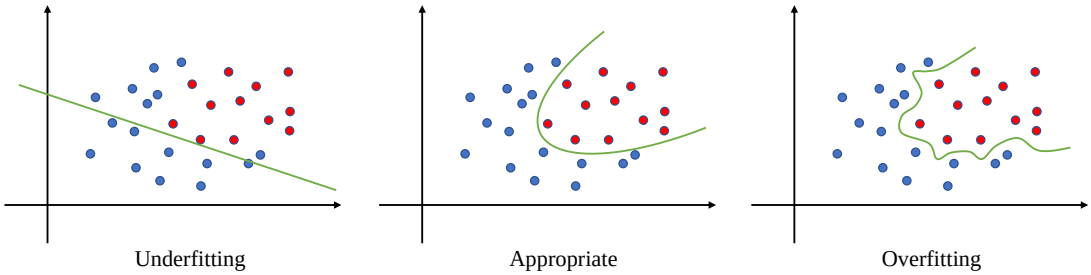


Fig. 3.12 Underfitting, appropriate and overfitting

3.3.4 Techniques in training neural networks

1. Early stopping:

It is common to build more complex deep neural networks than the problem really needs. In fact, it is a paradox to tell if the architecture of the neural networks fits the problem: the neural networks need to be trained before judging the complexity. Assume the neural networks that more complex than the problem needs are built. Fig. 3.13 shows the changes in the errors of the same neural networks in different training epochs in a points classification task.

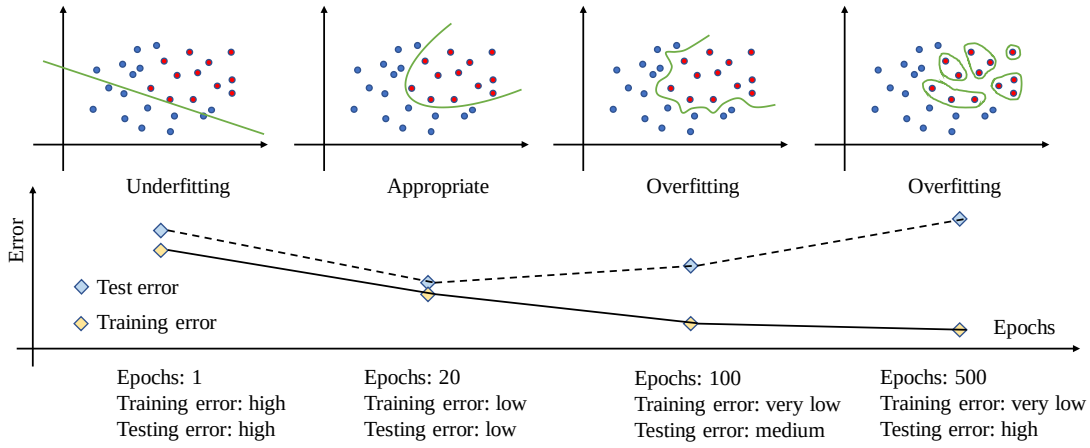


Fig. 3.13 Training epochs and errors

In Fig. 3.13, the training begins with random weights in epoch 1, in which both the training error and the testing error are high. The neural networks in epoch 1 are clearly underfitting that too simple to understand the problem. When the training process goes to epoch 20, both the training error and the testing error decrease and the neural networks are well trained for the classification problem. But with the training going on to epoch 100, the training error continuously goes down while the testing error begins to increase. Now the neural networks try to remember everything in the input data even there are noisy and fluctuations in them. The neural networks begin to make more

mistakes in the testing phase but still acceptable. When the training goes on to the epoch 500, there is vast differences between the training error and the testing error. The neural networks remember the inputs too clearly to do predictions flexibly. This is the training process that epochs effect the performances of the same model.

More generally, the relationships of training epochs and errors can be drawn as the figure shown in Fig. 3.14. The horizontal axis refers to the complexity of the model. In this case, the complexity is the number of epochs. The more the model is trained, the more the model learns and the more complex it becomes. The training error deceases with the increase of the epochs, which means that the model is learning more of the inputs through each epoch. While the testing error deceases to a minimum then rises again, which means that the model is functioning from underfitting to overfitting. The best performance of the model occurs at the lowest error point in the testing error. The epoch at this point is when the training should stop. There are methods to make sure the training process stops at this point and keep the best performance of the neural network, which is called early stopping.

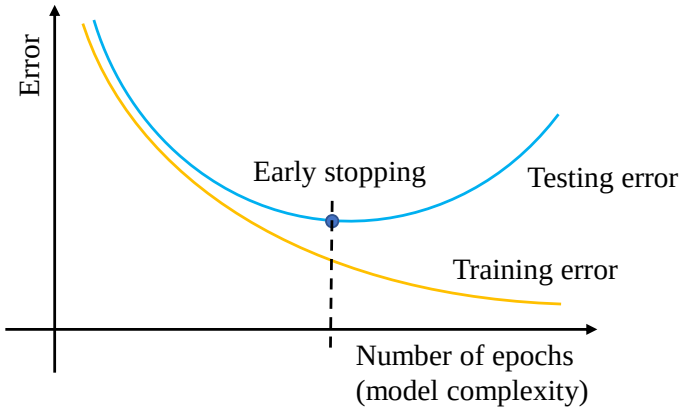


Fig. 3.14 Model complexity graph

2. Regularization:

Recall the linear prediction to one point (x_1, x_2) in 2.1:

$$y = \sigma(\omega_1 x_1 + \omega_2 x_2 + b)$$
$$\sigma: \text{ sigmoid function}$$

When the point (x_1, x_2) is (1,1), but the weights (ω_1, ω_2) are (1, 1) and (10, 10) respectively (b =0), the predictions to the same point are:

$$\sigma(1+1) = 0.88$$
$$\sigma(10+10) = 0.9999999979$$

The conclusion is: larger weights result in the tendency of overfitting. The predictions are closer to 1 if the weights are large even if the points are the same. This leads to a lower error but steeper active function which is harder to do gradient descent (shown in Fig. 3.15). The derivatives are nearly to 0 a little far from the central but very large near the central of the curve. It is better to generate a model with smaller weights like the curve on the left. The curve on the right is too certain to provide flexibility to apply gradient descent. The curve on the right would generate great fluctuates of errors because the predictions are closer to 1 or 0.

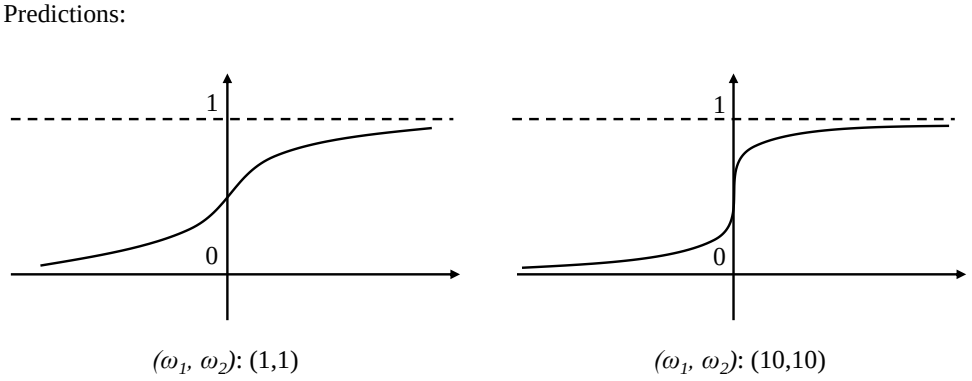


Fig. 3.15 Weights affecting the prediction

The solutions to the problem of overfitting induced by too large weights are to tune the error functions. The basic concept is to punish high weighs. By adding the old error function to the penalty of weights, the adjusted error functions can be expressed as:

L1 Error Function:

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(y_i) + (1-y_i)\ln(1-y_i) \right] + \lambda(|\omega_1| + |\omega_2| + \ldots + |\omega_n|) \quad 3.2$$

or L2 Error Function:

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(y_i) + (1-y_i)\ln(1-y_i) \right] + \lambda(\omega_1^2 + \omega_2^2 + \ldots + \omega_n^2)$$

The parameter λ is to determine how much the penalties to large weights are to adopt. The L1 error function takes the absolute of the weights to prevent from too small errors if the weights are too large and the L2 error function takes the squares of the weights respectively. Either of them is suitable for different models.

3. Dropout:

Another solution to prevent overfitting is dropout [123]. The training of neural networks begins with randomly assigned weights. There are tendencies that the larger the node is initialized, the larger it would become in the following epochs, and vice versa. The results to such phenomenon are that there will be "dead" nodes in the neural networks that were never well trained. The dropout method prevents these crippled nodes from occurring using the randomly "killing" method. As shown in Fig. 3.16, each node in one layer is multiplied by an independent Bernoulli random variable with the probability p of being 0, otherwise the variable is 1, before propagates into the next layer. The method of dropout may seem too radical for training, but the results show that it improves the accuracy. Using dropout can balance the nodes trained evenly. Dropout is also getting better results in neural networks than those not used.

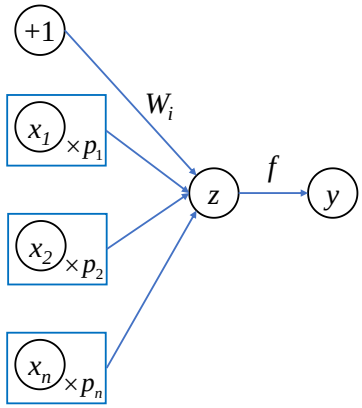
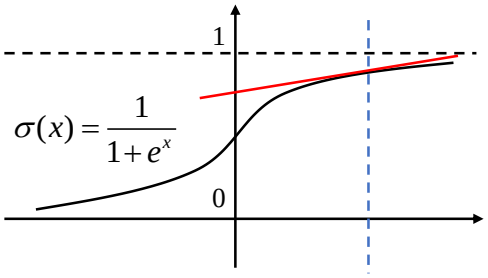


Fig. 3.16 Dropout

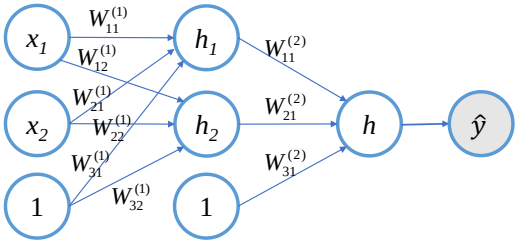
4. Vanishing gradient and other activation functions

In the backpropagation to the sigmoid function, when the input is far away from the zero point, the derivative is very small, almost to zero. The derivative is the opposite direction that the minimization occurs. When the derivative is too small it is slower or even never for the error function reaches its minimum. Things get even worse in the multilayer neural networks. For example, in the neural network shown in Fig. 2.10, the derivative of $\frac{\partial E}{\partial W_{11}^{(1)}}$ is calculated using the chain rule by multiplying several derivatives of sigmoid functions which are small. The final result of $\frac{\partial E}{\partial W_{11}^{(1)}}$ is tiny if there are many layers with the activation function as sigmoid function (shown in Fig. 3.17). This is the vanishing gradient that would impede the learning process of the

neural networks.



(a) Derivative to sigmoid function far from zero point

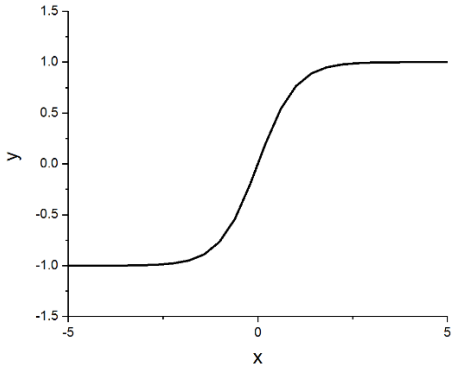


$$\frac{\partial E}{\partial W_{11}^{(1)}}=\frac{\partial E}{\partial y}\frac{\partial y}{\partial h}\frac{\partial h}{\partial h_1}\frac{\partial h_1}{\partial W_{11}^{(1)}}$$

(b) Tiny derivative

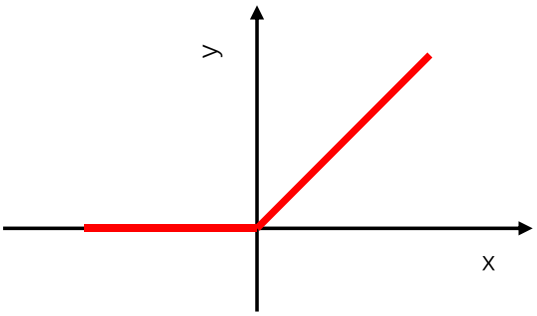
Fig. 3.17 Vanishing gradient

Changing the activation function can solve the problem of vanishing gradient. Bigger gradients of the activation functions are needed to prevent vanishing gradient in deep learning. Two popular activation functions that generate bigger derivatives are shown in Fig. 3.18. The ReLU function is used widely because it can improve the training significantly without sacrificing much accuracy because the derivative is 1 if the input is positive.



$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

(a) Hyperbolic tangent function



$$\mathrm{ReLU}(x)=\begin{cases}x \text{ if } x\geq 0\\ 0 \text{ if } x<0\end{cases}$$

(b) Rectified linear unit (ReLU)

Fig. 3.18 Hyperbolic tangent function and ReLU function

5. Local minimum, random restart and momentum

Recalling the error function shown in Fig. 3.9, the minimization usually faces with much more complex conditions that the error function is in an n-dimensional space. There are many local peaks and local minimums in the error function that would trap the minimization by gradient descent in the local minimums (Fig. 3.19). Gradient descent

is in a failure in such conditions.

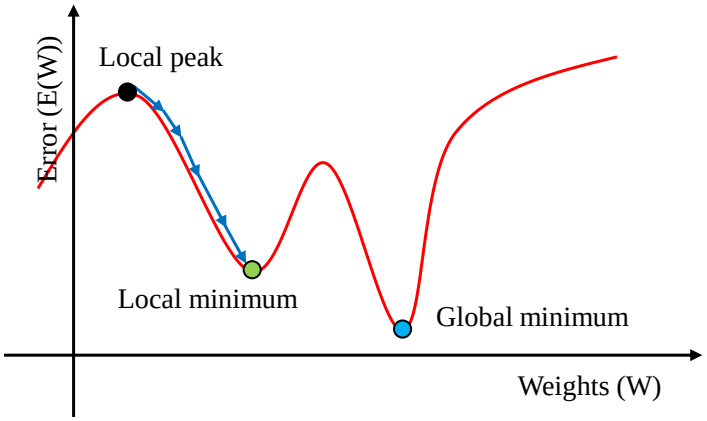


Fig. 3.19 Local minimum

One solution to the failure of gradient descent to local minimum is random restart (Fig. 3.20). The gradient descents are started from a few different random points that the number of these points conforms with the complexity of the error function. This method increases the probability of reaching the global minimum and at least to an acceptable local minimum. But this method costs longer time to compute and it is difficult to estimate a proper number of start points for the complexity of the error function.

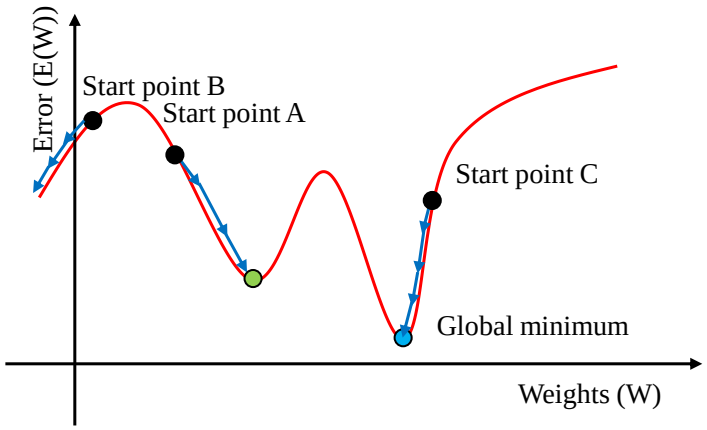
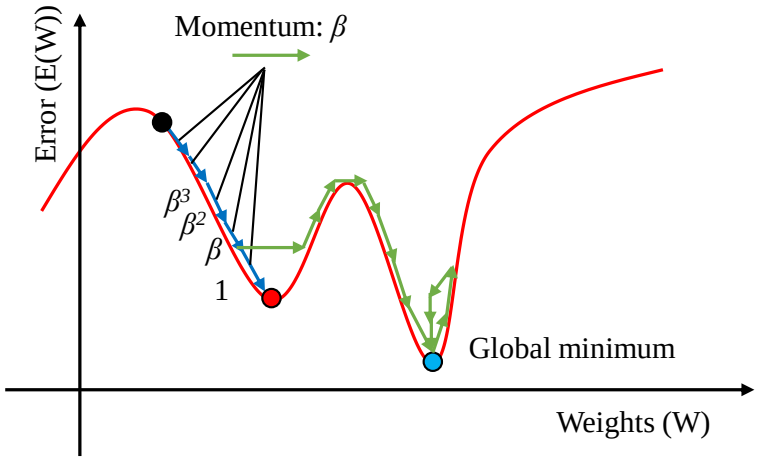


Fig. 3.20 Random restart

Another solution to local minimum is named momentum with the idea of introducing kinetic energy into steps in the gradient descent. The gradient descent gets power from a few previous steps and rushes over the local minimum to find a lower minimum (Fig. 3.21). Even though the step would rush over the global minimum for a few more steps, it will fall back to the global minimum because of not enough momentum. The coefficient of momentum is β, which is between 0 and 1. Each step is determined by the i steps previous to it. Although it is hard to prove its advantages in math, this

algorithm works well in practice.



$$STEP(n) = STEP(n-1) + \beta STEP(n-2) + \beta^2 STEP(n-3) + \ldots + \beta^i STEP(n-1-i)$$

Fig. 3.21 Momentum

6. Image augmentation

In machine learning, data is a valuable property to well training a model. Without enough well distributed data reflecting the real world, the model would not learn enough to make reliable predictions. However, data that reflecting the real-world problem are usually hard to collect, especially in highly specialized fields. There are two kinds of solutions to the lack of data: a more intelligent model that can learn from very few data and the data augmentation. In this thesis, the data are labeled ceilings images. Considering the characteristics of convolutional neural networks, adjusting the images will not undermine the translation invariance and representations in the images. For example, the size or the angle of the object will not affect the contents in the images. Moreover, augmenting the number of images by adjusting the images would make the convolutional neural networks more robust and better trained.

There are many ways in adjusting the image to generate more data (Fig. 3.22). All the generated images with the same label to the original images consist a much larger dataset for the neural network to learn.

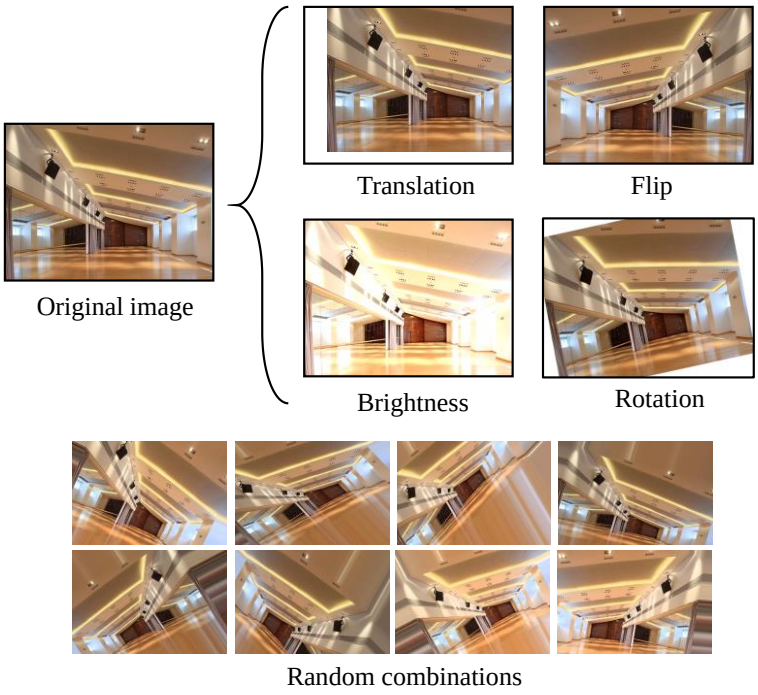


Fig. 3.22 Data augmentation

7. Cross validation: the training, validation and testing datasets

There should be dataset reserved for the performance verification of the trained CNN model. The simplest idea of splitting data is to divide the data into two sets: the training data and the testing data, which usually have the portion of 8 to 2 or 7 to 3. The train data is used in the training process and should never be used to evaluate the performance of the model. The test data is used to evaluate the model after trained. For more complex problems, it would be problematic if the data is only split into two sets because the training process also needs partial evaluations during the training process. When such evaluations are not the final evaluation to the model, it is the start of overfitting. That is because when adjusting the model parameters due to the results from the training process evaluations, the directions that parameters should change are directed by the validation data in an inexplicit way. In other words, the validation data is also involved in the training process that makes the final evaluation to the model is unreliable.

The solution to such problem is to divide the data into three sets: the training data, the validation data and the test data. The test data is hidden from the model until the final evaluation to the model. The validation involves in the training process of the model. The ratio to these data sets are alterable which is usually 6: 2: 2. The most defect of splitting data set into three subsets is that it reduces available data for training.

8. Sensitivity and specificity

The accuracy is usually calculated as the percentage of correctly predicted inputs divided by the whole inputs: $Accuracy = \frac{True}{True + False}$. The terms of sensitivity and specificity are usually used to evaluate a clinical test, which is a typically binary classification test [124]. In the case of evaluating the accuracy of deep learning models for binary classification tasks such as ceiling damage evaluation, the sensitivity and specificity are introduced to better analysis the deep learning model from the perspective of accuracy. Two classes of ceiling images exist with the contents of intact and damaged ceilings, and two kinds of predictions to one input image: intact and damaged. Therefore, there are four combinations to the contents and the predictions:

1. True positive: the ceilings in the image are 'damaged' and the prediction is 'damaged'.

2. False positive: the ceilings in the image are 'damaged' but the prediction is 'intact'.

3. True negative: the ceilings in the image are 'intact' and the prediction is 'intact'

4. False negative: the ceilings in the image are 'intact' but the prediction is 'damaged'.

Wrong predictions (false positive and false negative) are not welcome in most cases and they are where efforts spent on to prevent. In these two kinds of wrong predictions, the most unfavorable condition is when the input itself is problematic but the prediction to it fails to detect it. In the case of ceiling damage evaluation and detection, the most dangerous situation is that the ceilings are damaged while the ceiling damage detection approach fails to recognize and reports no alarm. Based on the four terms above, factors evaluating the predictions are:

1. Sensitivity measures the proportion of true positives to those who are really in the positive situation by $Sensitivity = \frac{True\ positivities}{True\ positivities + False\ negativities}$;

2. Specificity measures the proportion of true negatives to those who are really in the negative situation by $Specificity = \frac{True\ negativities}{True\ negativities + False\ positivities}$.

Both the sensitivity and the specificity are the higher the better.

3.3.5 Training the convolutional neural networks model

There are many adjustable parameters and hyperparameters in the training process. The architecture of the CNN must be determined before training. It includes the overall arrangement of layers in the model, the details of each layer. The architecture also should consider the complexity of the problem it deals with, like the input characteristics, the output requirements and the possible variations of the problem and dataset in the future. By combining and attempting the techniques above, a relatively good-performance convolutional neural networks model is obtained. The convolutional neural networks are built based on the architecture shown in Fig. 3.8. The details of the convolutional neural networks are shown in Table 3.1. There are 22 layers and 697,025 trainable parameters in the convolutional neural networks. All the images are preprocessed to the dimension of 400x600x3 before sent into the input layer. The whole data set (1953 images) is divided into train, validation and test sets with the ratio of 6: 2: 2 (shown in Table 3.2).

Table 3.1 Details of the convolutional neural networks

Layer		Input Shape	Output Shape	Parameters
1	Convolutional Layer (32 filters of (2, 2) size, stride: (1, 1))	(Batch size, 400, 600, 3)	(Batch size, 399, 599, 32)	416
2	ReLU Activation Layer	(Batch size, 399, 599, 32)	(Batch size, 399, 599, 32)	0
3	Max Pooling Layer (pool size: (2, 2), stride: (2,2))	(Batch size, 399, 599, 32)	(Batch size, 199, 299, 32)	0
4	Convolutional Layer (32 filters of (2, 2) size, stride: (1, 1))	(Batch size, 199, 299, 32)	(Batch size, 192, 298, 32)	4128
5	ReLU Activation Layer	(Batch size, 192, 298, 32)	(Batch size, 192, 298, 32)	0
6	Max Pooling Layer (pool size: (2, 2), stride: (2,2))	(Batch size, 192, 298, 32)	(Batch size, 99, 149, 32)	0
7	Convolutional Layer (32 filters of (3, 3) size, stride: (1, 1))	(Batch size, 99, 149, 32)	(Batch size, 97, 147, 32)	9248
8	ReLU Activation Layer	(Batch size, 97, 147, 32)	(Batch size, 97, 147, 32)	0

9	Max Pooling Layer (pool size: (2, 2), stride: (2,2))	(Batch size, 97, 147, 32)	(Batch size, 48, 73, 32)	0
10	Convolutional Layer (32 filters of (3, 3) size, stride: (1, 1))	(Batch size, 48, 73, 32)	(Batch size, 46, 71, 32)	9248
11	ReLU Activation Layer	(Batch size, 46, 71, 32)	(Batch size, 46, 71, 32)	0
12	Max Pooling (pool size: (2, 2), stride: (2,2))	(Batch size, 46, 71, 32)	(Batch size, 23, 35, 32)	0
13	Convolutional Layer (64 filters of (3, 3) size, stride: (1, 1))	(Batch size, 23, 35, 32)	(Batch size, 21, 33, 64)	18496
14	ReLU Activation Layer	(Batch size, 21, 33, 64)	(Batch size, 21, 33, 64)	0
15	Max Pooling Layer (pool size: (2, 2), stride: (2,2))	(Batch size, 21, 33, 64)	(Batch size, 10, 16, 64)	0
16	Dropout Layer (0.5)	(Batch size, 10, 16, 64)	(Batch size, 10, 16, 64)	0
17	Flatten Layer	(Batch size, 10, 16, 64)	(Batch size, 10240)	0
18	Dense Layer	(Batch size, 10240)	(Batch size, 64)	655424
19	ReLU Activation Layer	(Batch size, 64)	(Batch size, 64)	0
20	Dropout Layer (0.5)	(Batch size, 64)	(Batch size, 64)	0
21	Dense Layer	(Batch size, 64)	(Batch size, 64)	65
22	Sigmoid Activation Layer	(Batch size, 1)	(Batch size, 1)	0
Total trainable parameters: 697,025 Batch size: the number of images as input in each step				

Table 3.2 Image numbers in datasets split for the CNN model			
Dataset	Label		Total
	0	1	
Train	687	483	1170
Validation	230	161	391
Test	230	162	392
Total	1147	806	1953

The training runs on the hardware of GPU: Nvidia GeForce GTX1080Ti. In each epoch, the number of training step is 125, within which each step computing the batch size of 16 and the validation step is 50 with the batch size of 16. The batch size refers to the number of images used for training in one iteration. There are 30 epochs in the training process. It takes about four hours to train the neural networks. Images in both the train set and validation set are augmented randomly. The details of the randomness of the augmentation are shown in Table 3.3. The randomness of alternations to the training data is severer than that to the validation data. The weights that has the lowest validation loss are preserved as the final CNN model weights. The accuracy and loss curves of the training and testing in each epoch are shown in Fig. 3.23. The tendency of the accuracy is that it increases with the epoch increases and the tendency of the loss is that it deceases with the epoch increases. The fluctuations in the curves are due to the randomness of the generated images by data augmentation. In the accuracy-epoch curve, the reason for the accuracy of the model to the training data is usually lower than that of the validation data is that the alternations in the training dataset is severer than that of the validation dataset, which indicates that the trained model is robust to the translation invariance. In the loss-epoch curve, the phenomenon that the loss of the validation dataset is usually lower than the training dataset is also due to different extents of alternations to the images. The weights are saved as the final model weights when lowest validation loss occurs at the 19th epoch (validation accuracy 85.9%).

The final saved weights composite the final trained model. By using the final trained model to predict the test dataset (never shown to the model during the training process and no data augmentation), the accuracy is 86.2%. To better investigate the accuracy, the true positive (both the image and the prediction are damaged); the true negative (both the image and the prediction are intact); the false positive (the image is intact while the prediction is damaged) and the false negative (the image is damaged while the prediction is intact) are shown in Table 3.4 with the whole test dataset of 392 images.

Table 3.3 Randomness of data augmentation to the datasets			
	Training data	Validation data	Test data
Horizontal Flip	Yes	Yes	No
Shear radians	– 0.2 ~ 0.2	– 0.1 ~ 0.1	0
Zoom in / out range	– 0.2 ~ 0.2	– 0.1 ~ 0.1	0
Rotation range (degree)	– 15 ~ 15	– 10 ~ 10	0
Width shift	– 0.2 ~ 0.2	– 0.1 ~ 0.1	0
Height shift	– 0.2 ~ 0.2	– 0.1 ~ 0.1	0
Brightness multiplier (HSV color)	0.5 ~ 1.5	0.5 ~ 1.5	1

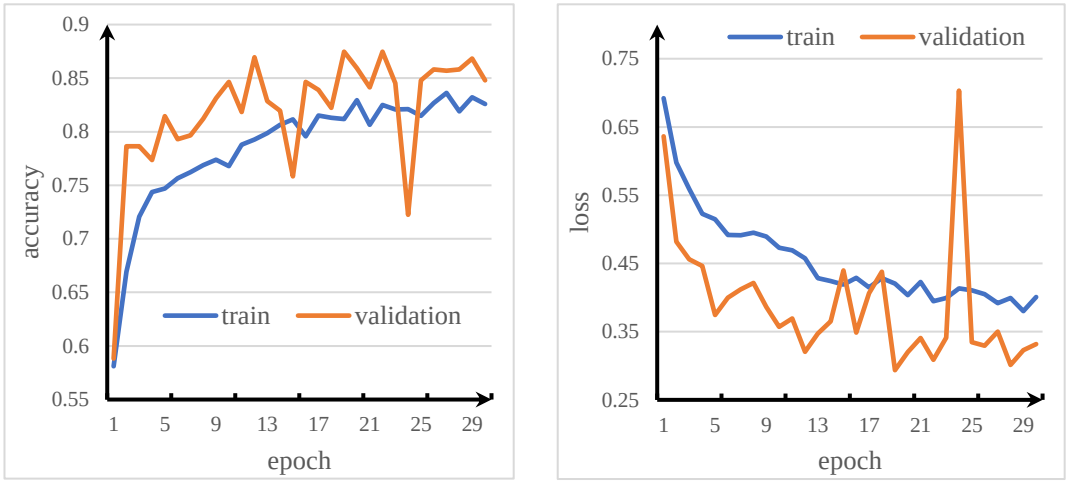


Fig. 3.23 Curves of training

Table 3.4 Sensitivity and specificity			
	Negative (0)	Positive (1)	Total
True	TN: 201	TP: 137	338
False	FN: 29	FP: 25	54
Total	230	162	392
Accuracy = (TP+TN) / SUM = 0.862 Sensitivity = TP / (TP+FN) = 0.825 Specificity = TN / (TN+FP) = 0.889			

3.4 Performances of the trained convolutional neural networks

To testify the trained CNN model, other ceiling images (both intact and damaged ones) are collected from the internet as the external ceiling image database. The basic requirements to the ceiling image collection are: (1) The main objects in an image are the ceilings. (2) The design of the ceilings is plain and not bizarre because the training dataset mainly contains regular ceilings. (3) Higher resolution images of ceilings are favorable.

All the collected ceiling images are resized to the resolution of 400x600x3 before evaluated by the trained CNN model. The ceiling images and predictions to them are shown in Fig. 3.24 (sources of the images are shown in Appendix A). Assume the boundary to the predictions between intact and damaged labels is at 0.5, which means that if the prediction to an image is over 0.5, the contents in the image are prone to show features of damaged ones with the label of damaged and vice versa. The larger the prediction is, the more dangerous the ceilings in the image are. Red subtitles indicate wrong predictions to the real contents in the images. The accuracy of the predictions is about 88.9% to the ceiling images from internet.

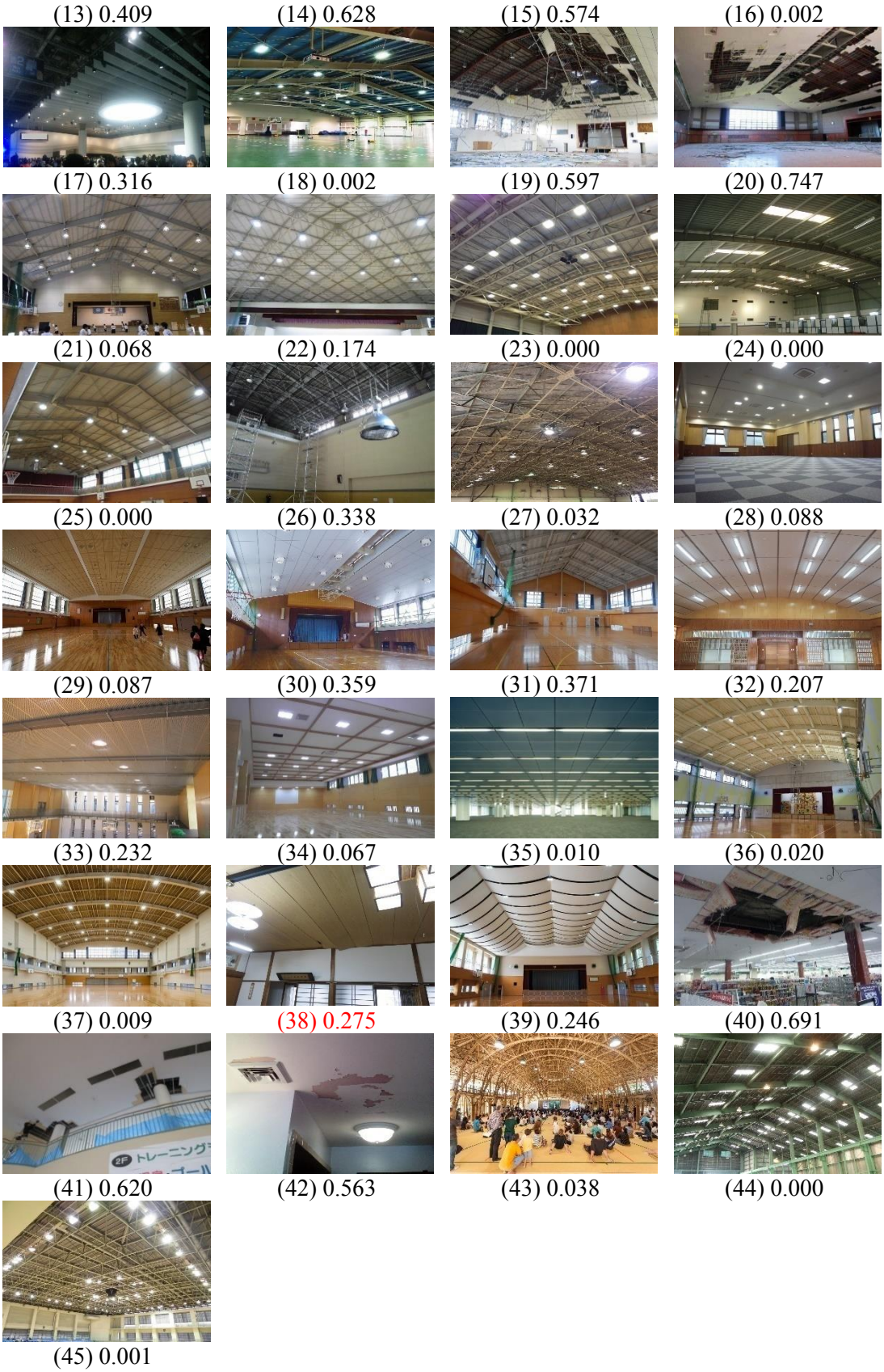


Fig. 3.24 Predictions to ceiling images from the internet

Although the accuracy to the images from internet is relatively high, there is improvement potential to the model. The dataset from internet collection is relatively small and only generates digital predictions from 0 to 1. It is hard to interpret the predictions and understand the trained CNN model only through the comparisons of original contents in the images and the predictions to them.

3.5 Conclusion

In this chapter, the main contribution is to apply convolutional neural networks in ceiling damage evaluation to fulfill the requirements for the deep learning model proposed in Chapter 1. A series measures are adopted to make sure that the CNN model really learns from the training dataset. The conclusions of this chapter are as follows:

1. The architecture of the CNN model for ceiling damage detection (shown in Fig. 3.25) is proposed to perform 3-2-1-dimensional reduction to the input image. The architecture is validated by the accuracy of 86.2% using testing dataset. Moreover, it is also confirmed that the CNN model can make acceptable prediction accuracy to the ceiling images collected from the internet (88.9%).

2. It is elucidated that the deep learning (convolutional neural networks) method is possible to be applied in ceiling damage evaluation even if: a. only two classes representing ‘intact’ and ‘damaged’ (images labeled by the same label can be totally different in manifestation), b. lack of training data (approximately only 1000 images for each class); c. high resolution of training data (400×600×3, to keep enough information).

3. Using high resolution images and scarce training data leads to overfitting. To overcome it, countermeasures such as data augmentation, cross validation and stochastic gradient descent are necessary to make the CNN models predict higher accuracies.

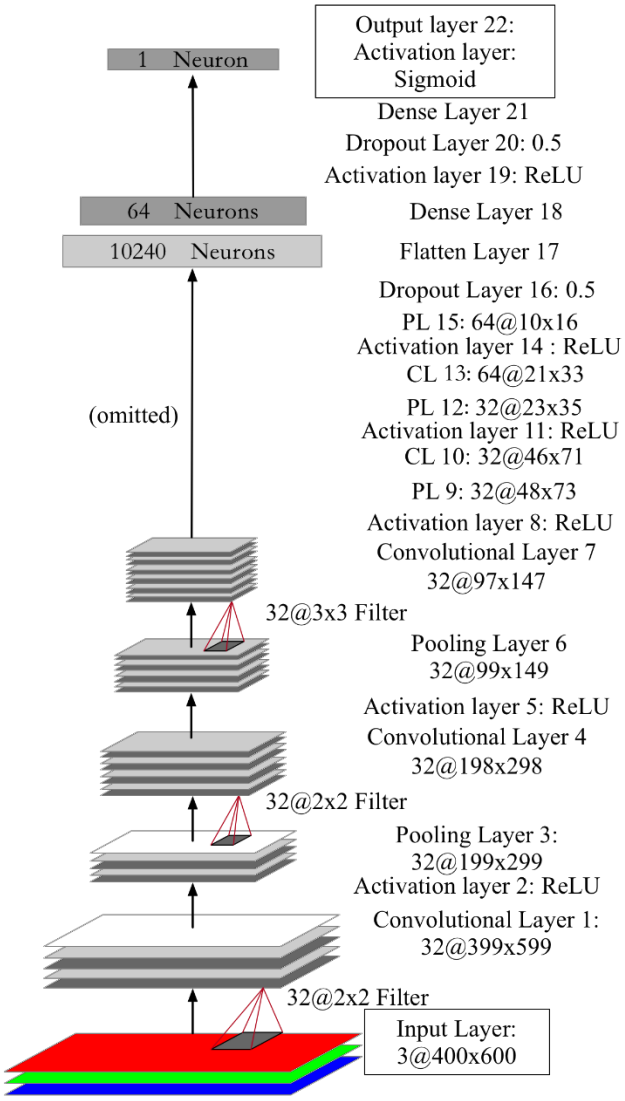


Fig. 3.25 The architecture of the CNN model for ceiling damage evaluation

4. Interpretations to the CNN Model by Visualization and a Ceiling Damage Detection System with User-CNN Interactive Process

This chapter is the core of the whole thesis. Through the visualizations to the trained CNN model, the CNN model is more understandable to human and no longer a black box. Furthermore, the visualizations of the saliency map and the Grad-CAM make ceiling damage detection possible. At the end of this chapter, a ceiling damage detection system is raised with the user involved in to perform ceiling damage detection both in the routine ceiling inspection by the maintenance personnel and safety confirmation to refugee under disaster circumstance.

4.1 Visualizing the layers and the patterns that maximize the activation feature maps

4.1.1 Visualizing the convolutional layers in the trained neural networks

As shown in Fig. 3.8 and Table 3.1, there are five convolutional layers in the CNN model. Each convolutional layer has its own filter parameters (filter numbers, sizes and strides). The parameters of the filters are optimized through the training process. The accuracy-epoch curve and the loss-epoch curve indicate the performance of the whole CNN model by mathematical calculations. Visualizing the convolutional layers has provided abundant understandings to the CNN architecture of deep learning [120, 125].The convolutional layers perform gradual abstractions to the information in the images. Each convolutional layer has its own structure that determines the qualities and sizes of filters. However, it is hard for human to interpret the CNN model only through the matrices in the layers of the final trained model. For example, the weight matrix of Convolutional Layer 1 is the combination of a four-dimensional matrix (2x2x2x32) and a one-dimensional bias matrix (1x32). These matrices are meaningless to human before they get applied to an image as filters. These matrices of filters can be considered as tools. It is hard to figure out how a tool really works only by looking at its appearance.

By visualizing the outputs from the input scanned over by each filter in a convolutional layer, it provides intuitions to see what parts in the image most activate the convolutional layer. If the CNN model is well trained, the intermediate outputs would gradually focus on the parts that are most likely to be damaged. There have been research on visualizing the objects that weights most to the final outputs to perform some kind of object detection [126].

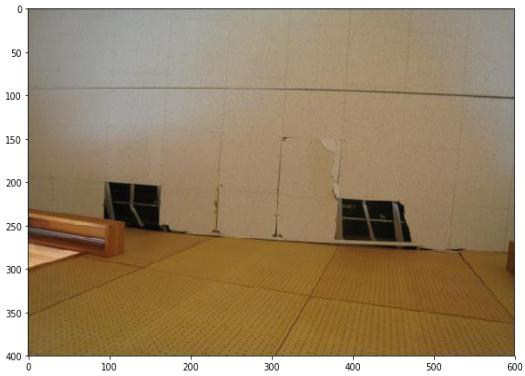
In the CNN model built and trained in this thesis, the output of the Convolutional Layer 1 is in the shape of (399x599x32), which indicates that there are 32 output images in the shape of (399x599x1). If the damaged parts are paid special attention to by the convolutional layers, it both indicates that the CNN model is in proper function and the activated parts are in high possibility of damage. The architecture of the CNN model in this research has five convolutional layers with 192 filters in total. Each filter scans over the three-dimensional input to it and outputs another three-dimensional output. All filters in one convolutional layer constitute the final three-dimensional output of the layer. Table 4.1 is extracted from Table 3.1 to indicate the details of convolutional layers (CL stands for convolutional layer).

Table 4.1 Convolutional Layers in the CNN model

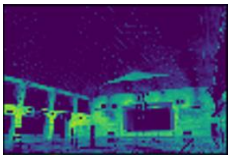
Layer	Input Shape	Output Shape	Filter Number and Shape	Parameters	Stride
CL 1	(400, 600, 3)	(399, 599, 32)	32 x (2, 2, 3)	416	(1, 1)
CL 4	(199, 299, 32)	(198, 298, 32)	32 x (2, 2, 32)	4128	(1, 1)
CL 7	(99, 149, 32)	(97, 147, 32)	32 x (3, 3, 32)	9248	(1, 1)
CL 10	(48, 73, 32)	(46, 71, 32)	32 x (3, 3, 32)	9248	(1, 1)
CL 13	(23, 35, 32)	(21, 33, 64)	64 x (3, 3, 32)	18496	(1, 1)

To inspect the outputs by the filters in the convolutional layers, eight images are chosen as inputs. Two of them are from the test data (never shown to the CNN model in the training). The other six are chosen from Fig. 3.24 (four of them are predicted correctly, two of them are predicted incorrectly). All the outputs (feature maps) to the 192 filters are shown in Appendix B. Fig. 4.1 shows some excerpts of feature maps in the intermediate convolutional layers. The name of "Fig. 4.1(a1) CL1_14" represents the feature map of the 14th filter in the CL1 layer.

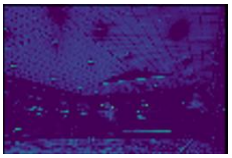
(a) An intact image from test data
Prediction: 0.006



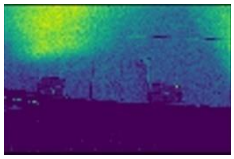
(b) A damaged image from test data
Prediction: 0.673



(a1) CL1_14

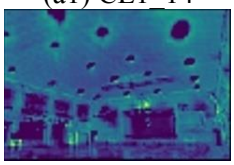


(a2) CL4_19

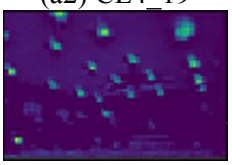


(b1) CL1_5

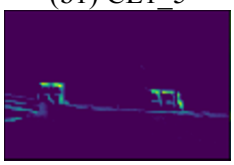
(b2) CL4_18



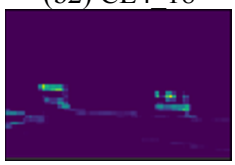
(a3) CL7_11



(a4) CL11_9



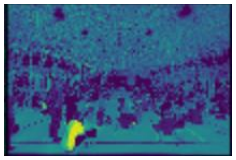
(b3) CL7_13



(b4) CL11_19

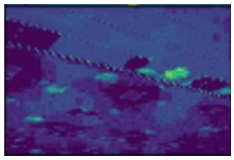
(c) Fig. 3.24(43)
Prediction: 0.039

(d) Fig. 3.24(5)
Prediction: 0.737



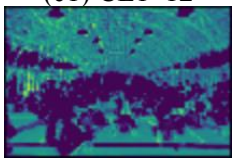
(c1) CL1_12

(c2) CL4_9

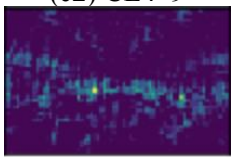


(d1) CL1_9

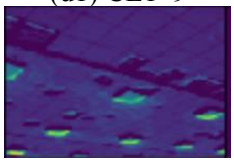
(d2) CL4_13



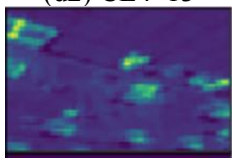
(c3) CL7_11



(c4) CL10_30



(d3) CL7_30



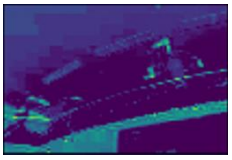
(d4) CL10_8

(e) Fig. 3.24(39)
Prediction: 0.246

(f) Fig. 3.24(41)
Prediction: 0.620

(e1) CL1_28

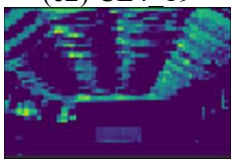
(e2) CL4_19



(f1) CL1_9

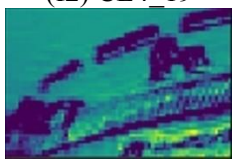
(f2) CL4_19

(e3) CL7_30



(e4) CL11_21

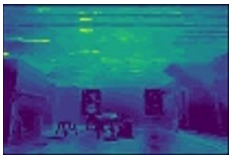
(f3) CL7_29



(f4) CL11_21

(g) Fig. 3.24(3)
Prediction: 0.502

(h) Fig. 3.24(38)
Prediction: 0.275

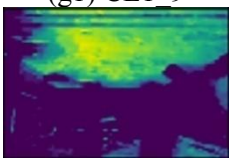


(g1) CL1_9

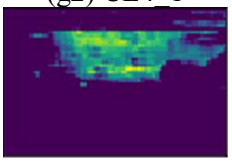
(g2) CL4_1

(h1) CL1_9

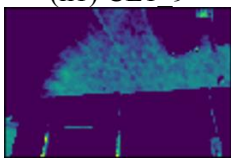
(h2) CL4_14



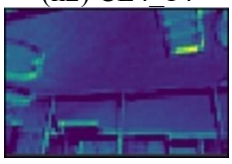
(g3) CL7_11



(g4) CL11_32



(h3) CL7_11



(h4) CL11_6

Fig. 4.1 Excerpts of the feature maps from eight images

The first two images in Fig. 4.1 are from the test data, in which the images are collected by the staff of the Kawaguchi Lab. Fig. 4.1(a) is a representative image of ceilings in a

stadium with large span. The feature maps can focus on the non-ceiling part and the ceiling part through (a1) CL1_14 and (a2) CL4_19. In the feature maps of (a3) CL7_11 and (a4) CL11_9, the filters focus on the interior without and with lightings. There are no obvious damages in Fig. 4.1(a), the CNN model does not pay special attention to anywhere that is suspected to be damaged. In Fig. 4.1(b), the damages are due to the earthquake. The earthquake shakes and squeezes the suspending system in the ceilings and results in the bending damage to the furring strips next to the wall. The bending and squeezing also makes the ceiling boards fall. The main damage form is the lack of ceiling boards. The feature map of (b1) CL1_5 indicates that the filter to the feature map distinguishes the wall and the ceilings, the feature map focuses on the ceilings. The feature maps of (b2) CL4_18, (b3) CL7_13 and (b4) CL11_19 focus on the damages in the zone of the missing ceiling borads.

The remaining six images in Fig. 4.1 are collected from the internet. Images of Fig. 4.1(c~f) are predicted correctly and those of Fig. 4.1(g, h) are predicted incorrectly by the trained CNN model. The feature map in (c1) CL1_12 shows that it focuses on the ceilings and floors while it ignores the crowd and the lightings in the ceilings. While (c2) CL4_9 focuses on the lights, the windows and the crowd in the contrast to those in (c1) CL1_12. (c3) CL7_11 focuses on the ceilings and the floors more that (c1) CL1_12 does. The final prediction to Fig. 4.1(c) is low that means the contents in it are intact. The damage form of the ceilings in Fig. 4.1(d) is the missing of ceiling boards and peeling offs of the ceiling board surfaces. The feature maps of Fig. 4.1(d) provides interprets of the final prediction of damaged ceilings. (d1) CL1_9 is most activated by the peeling off boards while (d2) CL4_13 is most activated by the missing ceiling boards, both of which are related to the damage label. (d3) CL7_30 is most activated by the lights in the ceilings, it also notices the dividing lines among the ceiling boards in the upper part of the image. (d4) CL10_8 is activated by both the missing boards and the lights, noticing that the contrast of these two items are the strongest in the image.

Fig. 4.1(e) shows the membrane ceilings with its inherent curved surfaces. There are no membrane ceiling images in the training or in the validation data sets. In fact, the shape of the curved ceilings was never shown to the CNN model before. The image of the membrane ceilings is chosen to investigate if the trained CNN model can make correct predictions to new forms of ceilings that it was never trained for. (e1) CL1_28 indicates that the model still can tell the differences between the celings and the floor (although the window zone is also activated with the ceiling zone). (e2) CL4_19 indicates that the curve shape (gaps between the membranes) of the menbrane ceilings is most activated. (e3) CL7_30 and (e4) CL11_21 focus on the surfaces of the membrane. The final prediction to Fig. 4.1(e) agrees with the contents. Fig. 4.1(f) shows the damages in the ceiling boards that surrounding the columns in an earthquake. (f1) CL1_9 highlights the half-fallen parts at the rims of the falling boards. (f2) CL4_19 and (f3) CL7_29

foucus on the edges of the missing parts in the ceilings, even if the vents of the air conditioning. (f4) CL11_21 extract the contours of the void boards and areas with strong contrast.

Predictions to Fig. 4.1(g) and Fig. 4.1(h) are where the CNN model fail. The ceiling image of Fig. 4.1(g) shows a wood board ceiling with patterns of spots, which look like rust or water stains. Maybe this is the reason why the CNN model predicts to the ceiling image incorrectly (the prediction is 0.502, which is hard to really judge if the prediction is correct). (g1) CL1_9 and (g2) CL4_1 notice the celing and other parts except the ceiling respectively. Notice that (g1) CL1_9 is more activated by the textures in the wood which look like water stains. (g3) CL7_11 and (g4) CL11_32 both focus on the ceiling part with different extents. It can be still redeemed as a success because the result that the CNN model predicts alarms the user to perform more careful examination.

The damage in Fig. 4.1(h) is hard to find for human even at the first glance. The uplift between the ceiling boards is tiny. There is no such damage form in the training data set either. (h1) CL1_9, (h2) CL4_14 and (h3) CL7_11 do focus on the ceiling part of the image, but they fail to detect the uplift part in the ceilings. (h4) CL11_6 just sees the ceilings as a whole, an intact whole. It also fails to detect any damage traces in the image. The final prediction to this image is 0.275, which is relatively low.

By visualizing the outputs of the intermediate layers, it is possible to understand what the CNN model does in the convolutional layers and what countermeasures should be taken. It also aids to find out if the CNN model does learn from the training data or it just predicts by luck. The visualizations above show that the trained CNN model does learn from the training data and the accuracy can be improved by collecting more versatile ceiling images.

4.1.2 Visualizing the patterns that most activate the hidden layers in the convolutional neural networks

Visualizing the outputs of the intermediate convolutional layers by showing them different images helps to interpret the trained CNN model. However, these interpretations are from the intuitional point of view and not unified guiding significance. They still leave the questions of choosing how many and what kind of images to the CNN model. It is good for the understandings if the images have something in common. But there are too many possible influence factors in the images. Note that we confined ourselves in the test data and internet ceiling data sets for looking for general input patterns. It is possible to take a further look at the trained CNN model.

Visualizing the matrices of the filters in the convolutional layers is no use, but how about generating an image that maximize the activations to the filters? In other words, how about taking the generating images task as an optimization problem?

The idea of generating an image that maximizes the activations to the filters dates back to the year of 2006. Hinton et al. [127] built a generative model that gives better digit classification than the best discriminative learning algorithms. The generated images form the deep hidden layers make the interpretations to the nonlinear, distributed representations possible. Erhan et al. [128] used activation maximization to visualize unsupervised deep learning models by gradient ascent in the images. Their results confirm that the higher layers in the deep learning model represent features more complicated than that the lower layers do. This method was employed to visualize the features learnt by an unsupervised auto-encoder [129]. The application of this method in the visualizing the features of classes [130] in the deep image classification CNN model trained on the ImageNet challenge dataset [71]. The deep dream project launched by Google also used the visualization of the features in the filters to create arts [131] as shown in Fig. 4.2. This idea of generating images from the features that the CNN model learned also revives the art historical research in iconography and formalism [132]. The visualization of intermediate layers and regularized optimization are introduced in helping to interpret the trained CNN models [133].

Fig. 4.2 Examples of Google's deep dream works [131]

This is the inverse problem to the training of CNN models. The training process is to generate the weights that minimize the loss function using data (images) that we have already obtained by gradient descent method. While when visualizing the patterns that most activate the hidden layers, the objective is to generate an image using the trained-already weights that most activate the filters. Corresponding to the gradient descent method, the gradient ascend method is adopted to fulfill the purpose of generating the

pattern images (Eq. 4.1).

Gradient descent (minimizing an objective function $J(\theta)$): $\theta_i \leftarrow \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$

Gradient ascent (maximizing an objective function $J(\theta)$): $\theta_i \leftarrow \theta_i + \alpha \frac{\partial J(\theta)}{\partial \theta_i}$ 4.1

where θ is the weights, $J(\theta)$ is the objective function, α is the learning rate

The idea of finding an image that maximizes the activations in the intermediate layers can be viewed as

$$X^* = \arg\max_X (h_{ij}(\theta, X)) \quad 4.2$$

where: θ is the weights and biases in the trained neural networks model (constant);

X^* is the image that maximizes the activation;

X is the input;

$h_{ij}(\theta, X)$ is the activation of a given unit i in a given layer j.

This is an optimization problem that can be done by performing gradient ascent in the input space X. In other word, this problem is to move X in the direction of the gradient $\frac{\partial h_{ij}(\theta, X)}{\partial X}$ until the local maximization X^* that maximizes $h_{ij}(\theta, X)$. This optimization strategy is applicable to any neural networks as long as the activation function $h_{ij}(\theta, X)$ can be computed. The optimization also involves the choice of learning rate and stop criteria like the gradient descent.

In the case of finding an image that most activates the filters in our trained CNN model for ceilings, the function can be specified as

$$X^* = \arg\max_X (a_{ij}(\theta, X) - R_\theta(X)) \quad 4.3$$

where: θ is the weights and biases in the trained neural networks model (constant);

$X^* \in \mathbb{R}^{H \times W \times D}$ is the image that maximizes the activation, *H=400, W=600, D=3*;

X is the input image;

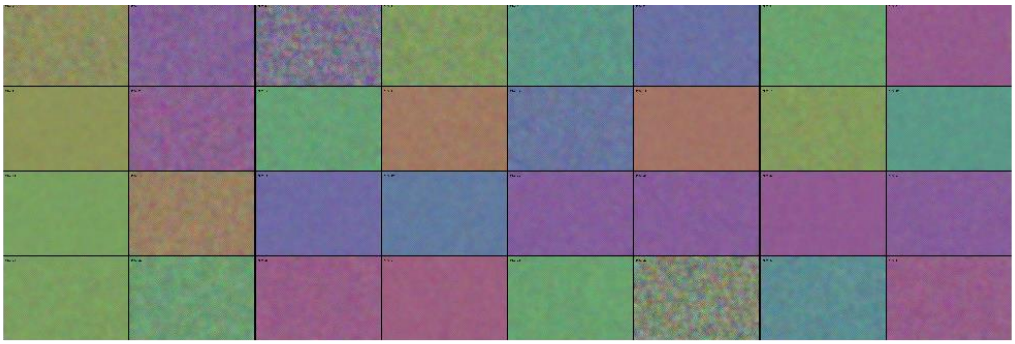
$a_{ij}(\theta, X)$ is the activation for a given filter i in a given layer j when the image is presented to the trained CNN model;

$R_\theta(X)$ is a regularization function that penalizes the $a_{ij}(\theta, X)$ function.

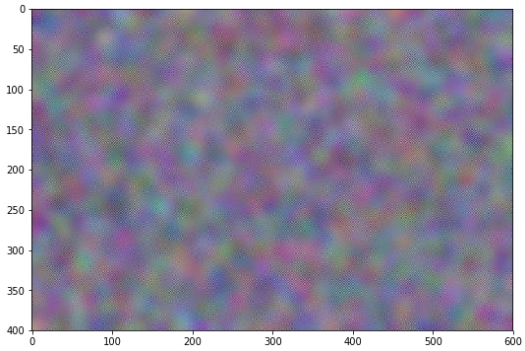
Just like the cycles we did in the gradient descent, the search for X^* starts from a random X_0, which is a random noised image. The step number and the learning rate are pre-fixed. Eq. 4.4 is a single step in the updating process:

$$X \leftarrow X + \alpha \frac{\partial}{\partial X}(a_{ij}(\theta, X) - R_\theta(X)) \qquad 4.4$$

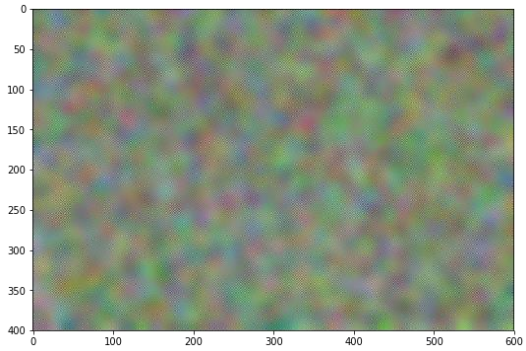
The maximized activations to the filters in the trained CNN model for ceilings are shown in Fig. 4.3. As shown in Table 4.1, the numbers of filters in the convolutional layers correspond to the maximized activations. For more clarity, two or more of the maximized activations in each convolutional layer are chosen to display. Notice that all the maximized activation images are in the shape of 400x600x3.



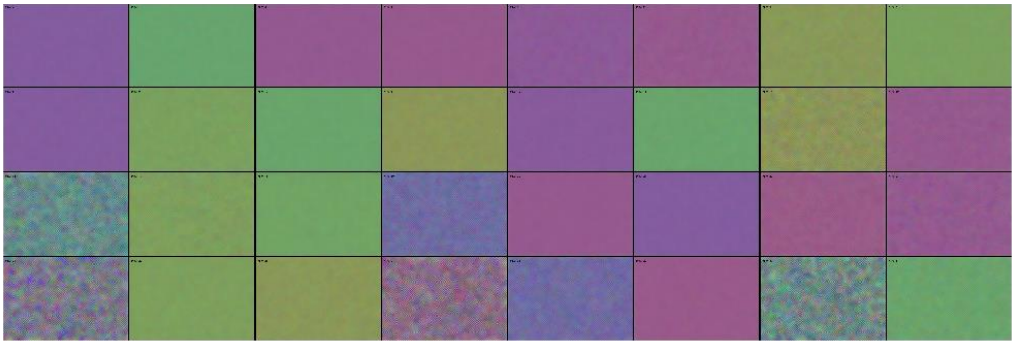
(a) CL1 (32 filters)



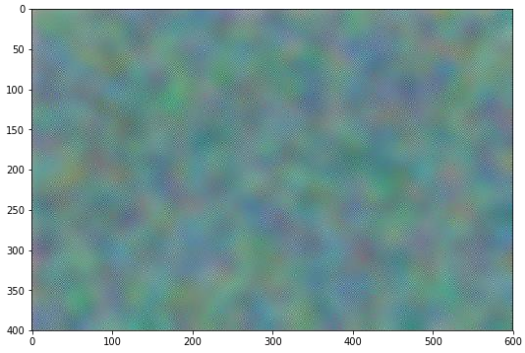
(a1) Filter 3



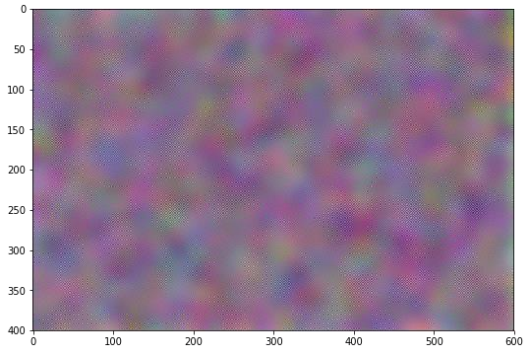
(a2) Filter 30



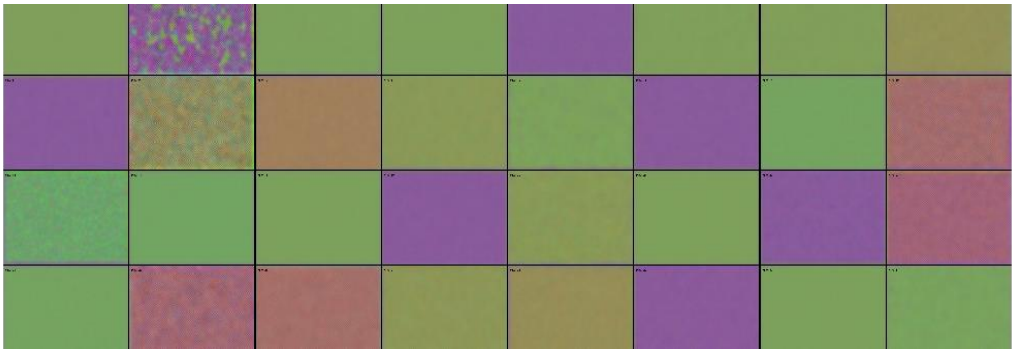
(b) CL4 (32 filters)



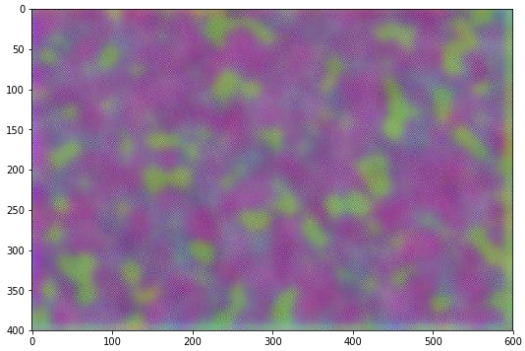
(b1) Filter 17



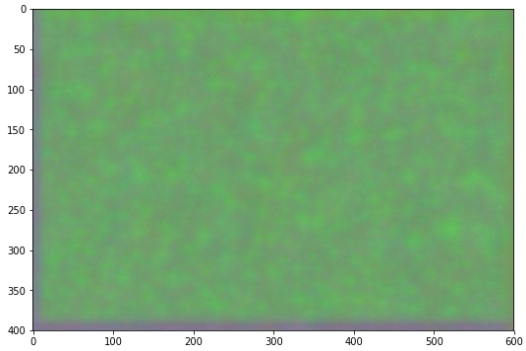
(b2) Filter 28



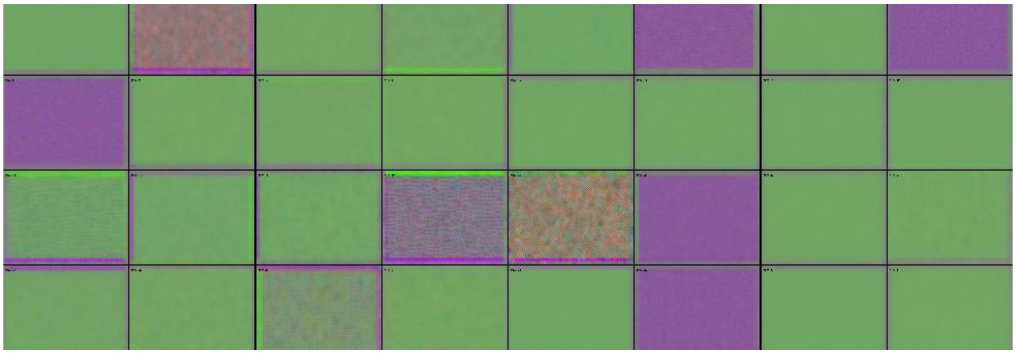
(c) CL7 (32 filters)



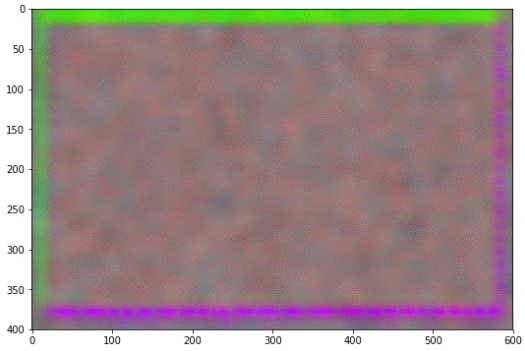
(c1) Filter 2



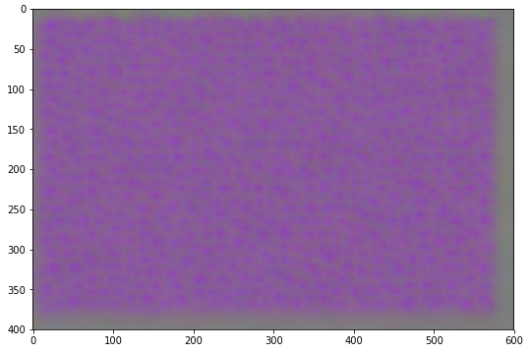
(c2) Filter 17



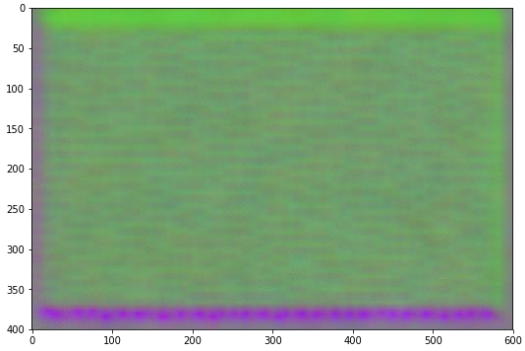
(d) CL10 (32 filters)



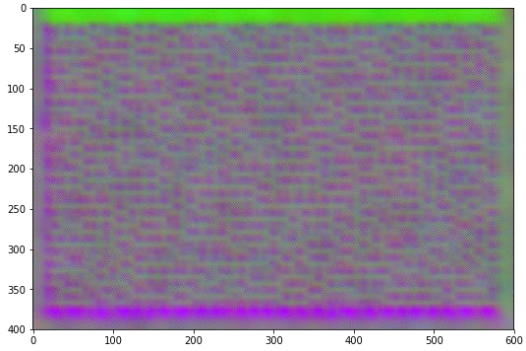
(d1) Filter 2



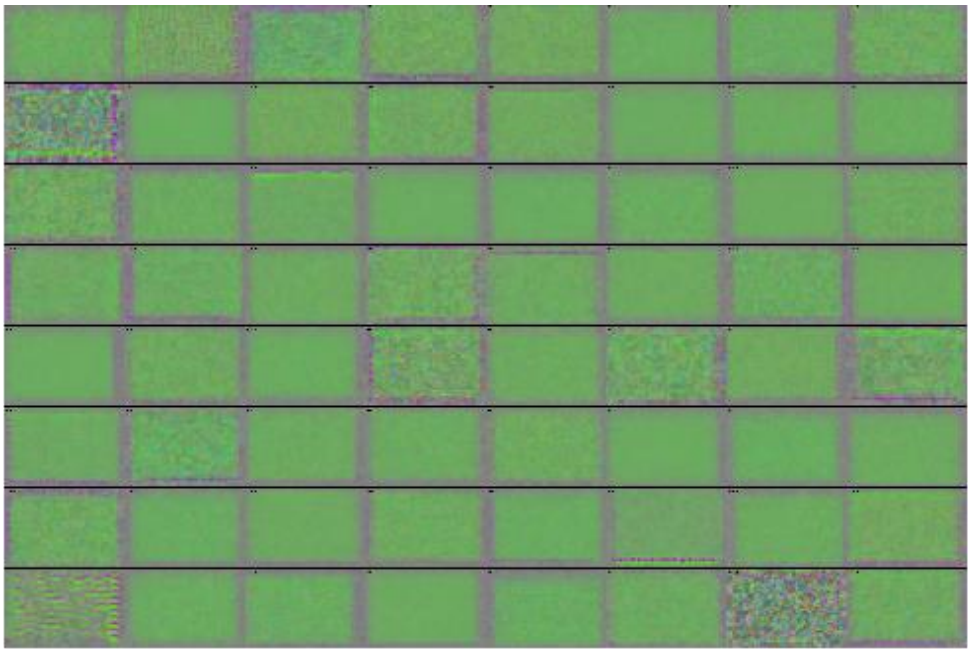
(d2) Filter 9



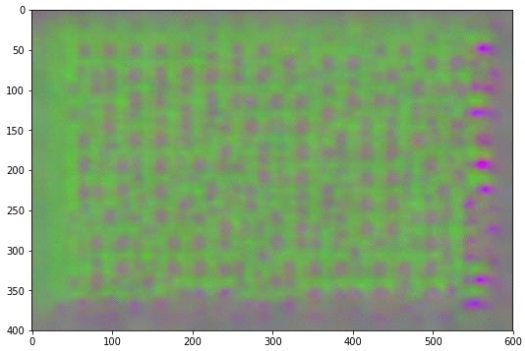
(d3) Filter 17



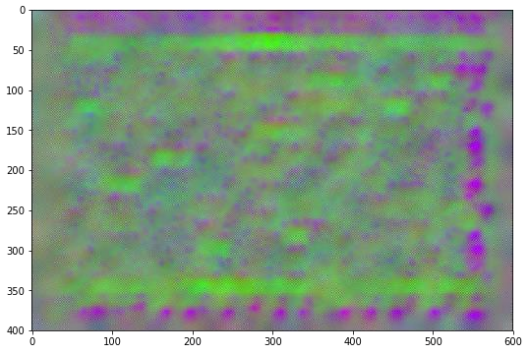
(d4) Filter 20



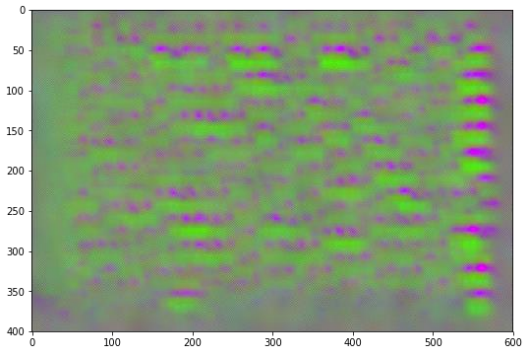
(e) CL13 (64 filters)



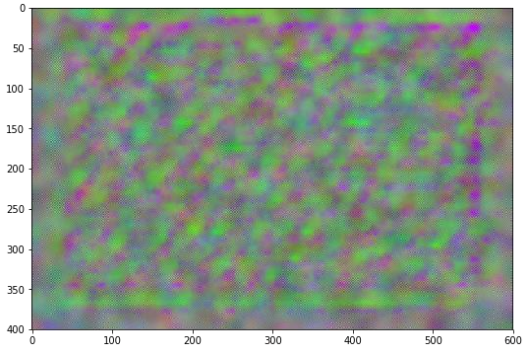
(e1) Filter 2



(e2) Filter 9



(e3) Filter 57



(e4) Filter 63

Fig. 4.3 Visualizations to the images that most activate the filters in the trained CNN model

From Fig. 4.3, the interpretations to the maximized activation images are:

(1) In CL1 and CL4, the filters do abstractions on randomly dispersed spots, in which there are no regular patterns among them.

(2) In CL7, some filters in it begin to be more activated by patterns looks like tissues (Fig. 4.3c1) or mesh connections (Fig. 4.3c2) from an organism.

(3) In CL10, the filters step further away that they are more activated by some regularly distributed shapes like evenly distributed points (Fig. 4.3d2) and Labyrinth-like grids (Fig. 4.3d3 and Fig. 4.3d4). Noticing the four edges in the maximized activation images in Fig. 4.3d, the pixels at the edges become different from the corner ones. That is due to the image augmentation in the training dataset: an original ceiling image was moved and rotated to generate more ceiling images which made the edges in the generated images meaningless (as shown in Fig. 3.22). The filters in CL10 noticed the meaningless zones and learnt to distinguish them.

(4) The abstraction in the final convolutional layer CL13 is stronger. The filters in CL13 combine the simple patterns into complex ones. Remember the final output to the CNN model is one node (from 0 to 1), which means the extent the contents in the input image is damaged to. Intuitively speaking, there are more common features in the intact ceiling images than those in the damaged ones. The filters in the CL13 are more activated by the images that have regular patterns. This indicates that the trained CNN model does learn the distinguishing methods by recognizing regular patterns.

(5) The abstraction is gradually stronger from the lower convolutional layers to the higher ones.

This section introduced a very powerful and useful method for visualizing the filters in the CNN model by finding an image that most activates a filter in a convolutional layer. It confirms the suspect that convolutional layers do gradual abstraction from the lower layers to the higher ones. It also provides direct inspections method to the filters in a convolutional layer to see if the trained model has really learnt something. Moreover, it helps with the determination to the architecture of the deep learning model that accuracy is not the only evaluation standard any more. People can find if or how much the deep learning model has learnt by visualizing the maximized activation images. It does open to black box of CNN models to some extent.

4.2 Damage detection

4.2.1 Saliency maps

Visualizing the middle convolutional layers outputs and visualizing the activation feature maps provide interpretations to the trained CNN model from the activation perspective. The expression of "activation" also connects to the phase "attention", which is more suitable to describe the artificial intelligence algorithms, especially the deep learning. However, visualizing the middle convolutional layers outputs and visualizing the activation features to the filters are not strong enough to provide strong guidance to human, in other words, we still do not know what regions in an image contribute most to the final predictions. This elicits the reflection of looking back into the trained deep learning models themselves. The first idea is simple: since the prediction accuracies to the contents in the image have been to very high levels by the deep learning models in classification task, why not investigate the prediction process to find out what parts in the image leads to the final prediction? The parts in an image that contribute most to the final prediction have the highest probability of being the target object. In other words, the object detection task is transferred into the task that given a trained classification model, the label to the target object and an image, we want to use them to perform localization to the given image for the given target object.

The visualization of image-specific class saliency maps was first introduced in 2014 [130], it is another powerful auxiliary means to investigate the trained deep learning model and can be modified for object detection. In the case of ceiling damage evaluation task in this thesis, it is important to confirm what parts in the ceiling image most activate the CNN model. Remember the final output is a float number from 0 to 1, which is impossible to interpret if the CNN is using ceiling-related pixels or using irrelevant pixels with ceilings like the windows or the crowd. The saliency maps solution kills two birds with one stone: 1. It confirms if the trained CNN model really learns the representative features of the objects; 2. If the trained CNN model really learns the representative features of the objects, it can do localization task to new images.

For example, a pretrained CNN model proposed in 2014 with the name VGG16 [134], whose team won the first and the second places in the localization and classification tracks respectively in ImageNet Challenge 2014 [71] with the database over 14million images to 1000 classes, achieved the accuracy of 93.2% in the top-5 test. The prediction to one image in classification is a 1000x1 vector with each row represents one class. We choose the 671st label in the classification index, which represents "mountain bike"

for the demonstration for saliency map method. Three images are downloaded from https://www.pexels.com under the CC0 license (https://www.pexels.com/photo-license/), in which one is a "mountain bike", one is a "scooter" and one is a "car". The original images and the saliency maps to them are shown in Fig. 4.4. It can be found that the saliency map to the "mountain bike" image is the strongest, the saliency map to the "scooter" is less strong and the saliency map to the "car" is null. The saliency maps confirm that the VGG16 does learn the features in the mountain bike and the saliency maps can be used for object detection.

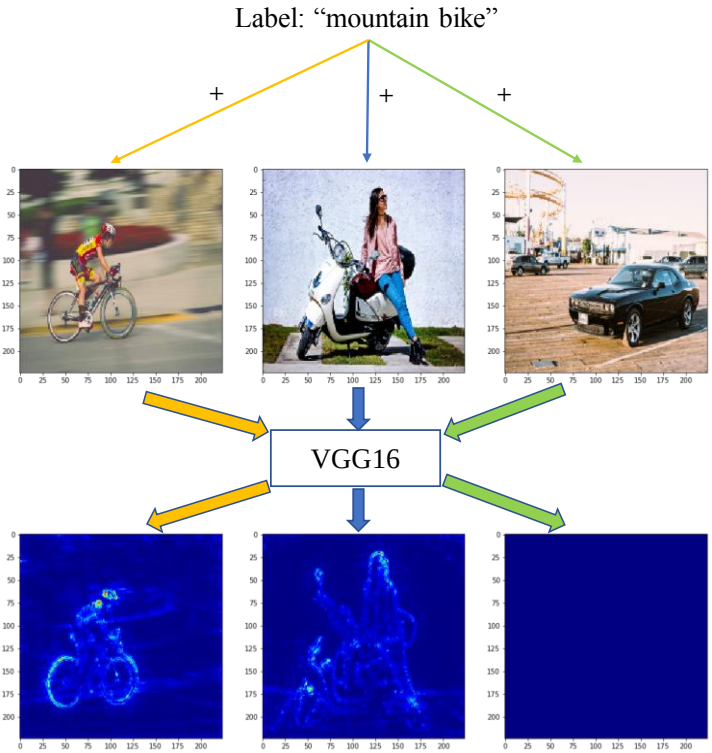


Fig. 4.4 Saliency map examples for VGG16

For a given image I_0 ($H \times W \times D$), a class with the index of c (n classes in total, the index is also the label to the object), and a trained CNN model, the final prediction to the image I_0 is a vector:

$$[S_1(I_0),\ S_2(I_0),\ \ldots,\ S_c(I_0),\ldots,\ S_n(I_0)] \tag{4.5}$$

$S_i(I_0)$ represents the probability that the image I_0 belongs to the i-th class, so the sum of them equals to 1:

$$\sum_i S_i(I_0) = 1 \tag{4.6}$$

By the way, the top-1 accuracy is the accuracy that the image label happens to be the highest probability prediction label; the top-5 accuracy is the accuracy that the image label happens to be one of the five highest probability prediction labels.

The goal is to identify which pixels in the image I_0 contribute most to the final prediction $S_c(I_0)$ that represents the class index c. The required output (the saliency map) is an image with the same shape to the given image I_0, which means that the input and the output are pixel-wise one-to-one correspondence. The relationship between the input I_0 and the saliency map to it can be represented as:

$$S_c(I_0) = \sum \omega_c^T I_0 + b_c \tag{4.7}$$

where ω_c is the saliency map to the image I_0.

In the case of CNN model, the final prediction corresponding to the class c is:

$$S_c(I_0) = f_1 f_2 \cdots f_n(I_0) \tag{4.8}$$

where f_i is the i-th layer in the trained CNN model.

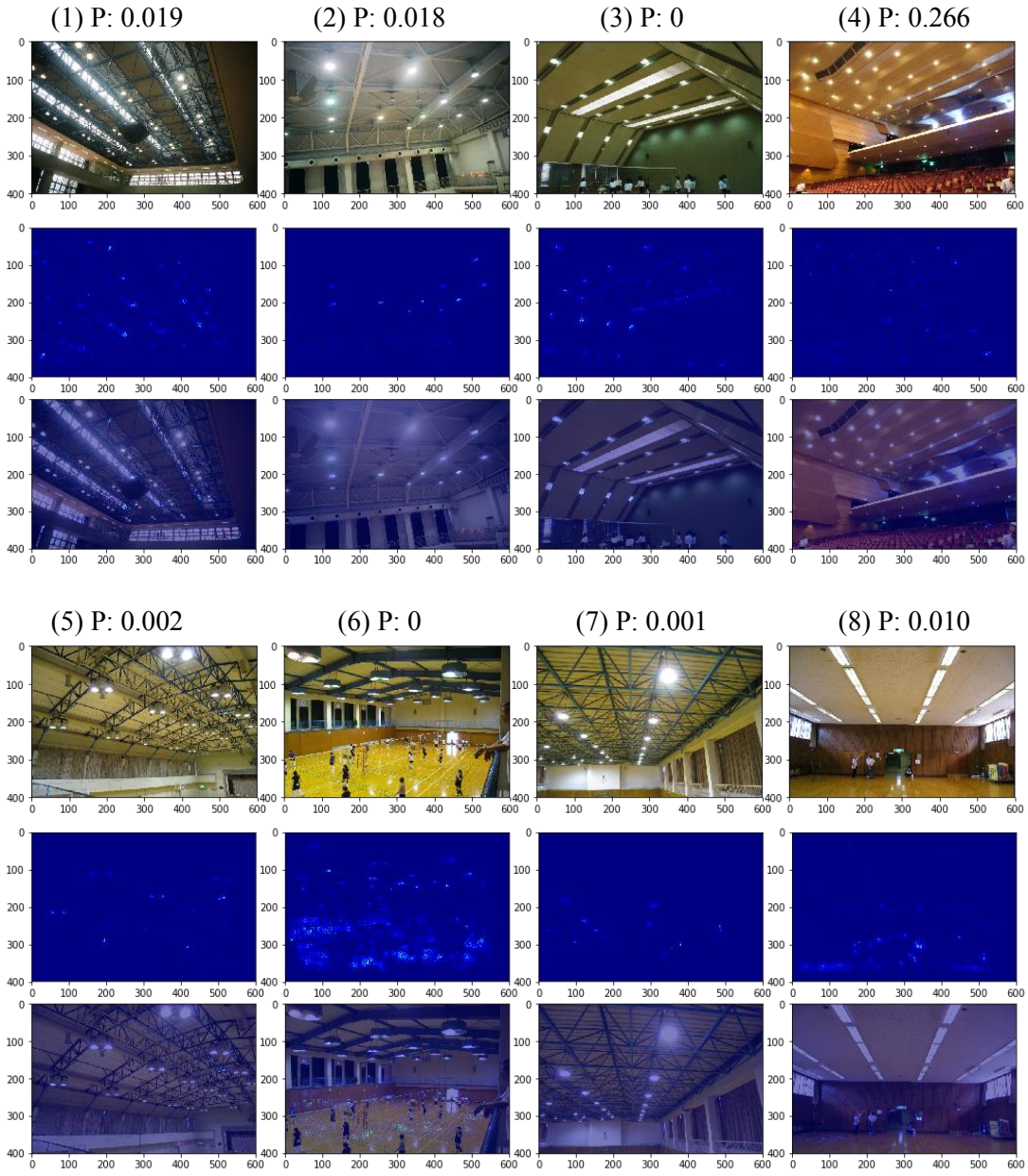
Unluckily, $S_c(I_0)$ is a multilayer non-linear function with respect to I_0, we cannot simply apply Eq. 4.8 to Eq. 4.7. Recall the first-order Taylor expansion:

$$S_c(I_0) = f_1 f_2 \cdots f_n(I_0) \approx \sum \omega_c^T I_0 + b_c \tag{4.9}$$

Now we can calculate the ω_c, both the saliency map to image I_0 and the derivative of $S_c(I)$ with respect to the input I at the point I_0:

$$\omega_c = \left. \frac{\partial S_c}{\partial I} \right|_{I_0} \tag{4.10}$$

Recall the backpropagation of deep learning, ω_c can be quickly computed by a single backpropagation. For the CNN model for ceiling damage evaluation in this thesis, the final output is only one class that represents the probability that the contents in the image are damaged. Applying the saliency map method to some images from the test dataset (shown in Fig. 4.5) and images from internet (from Fig. 3.24, shown in Fig. 4.6) with the final class index and the trained CNN model in Fig. 3.8. Three sub-images are shown for each ceiling image in columns, with the order of original-saliency map-overlaying the saliency map to the original image.



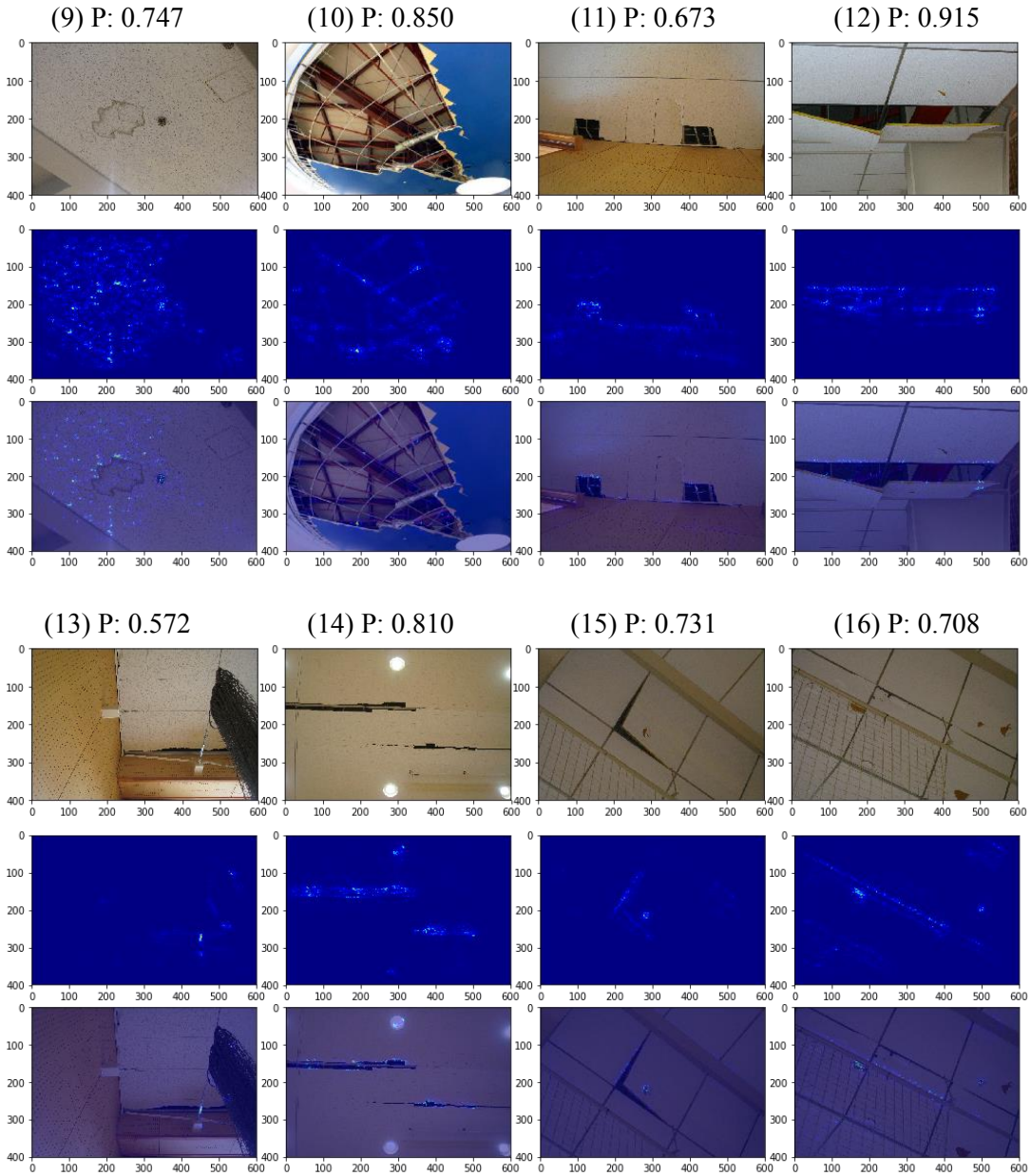


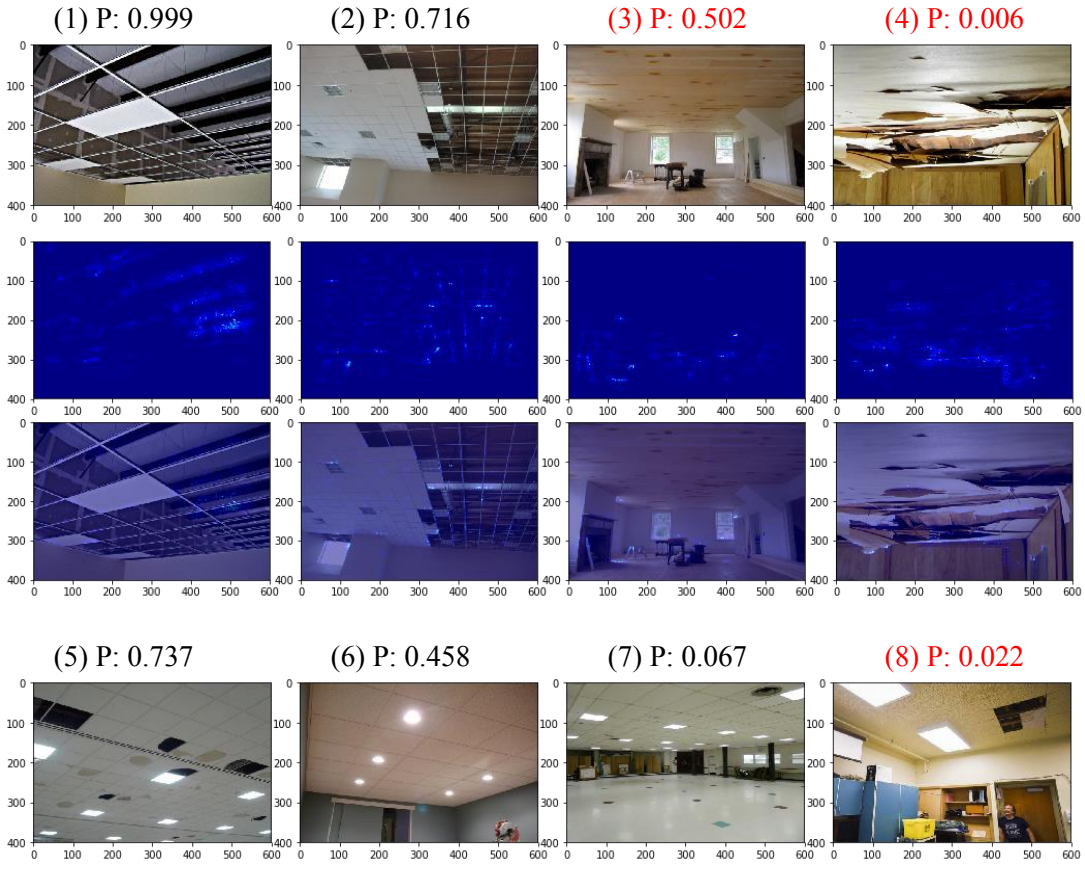
Fig. 4.5 Saliency maps to some ceiling images from the test dataset

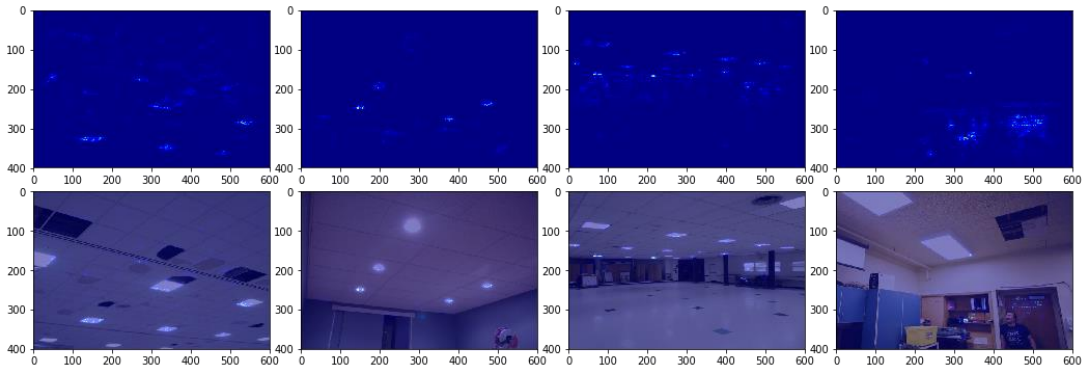
From Fig. 4.5, the conclusions to the saliency maps to the ceiling images from the test dataset are:

1. In the intact ceiling images, whose final predictions are very low, the most activated pixels are not ceiling-relevant while relating to lights and crowds. Noticing that the lights are usually well ordered and the crowds are extremely random. Although all of them are noticed by the CNN model, the contributions by these pixels to the final predictions are very low. It can be inferred that the model learns that the lights and the crowds are irrelevant to the predictions.

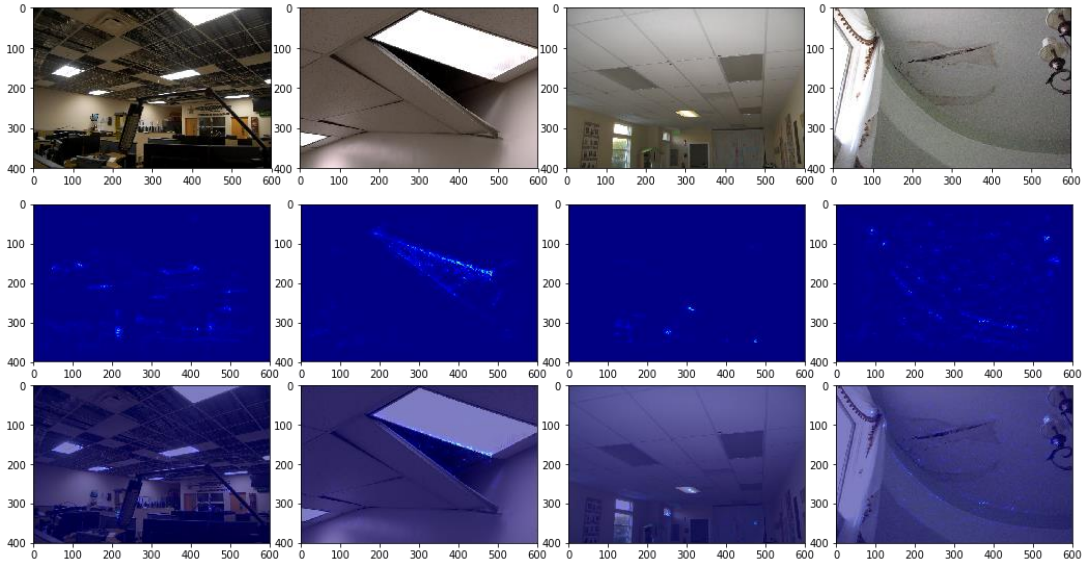
2. Results are different for the damaged ceilings. Firstly, the predictions to the damaged ceiling images are correct to the contents, indicating that the activated pixels in the saliency maps have strong weights to the final predictions. The shapes of the activated pixels are different from those in the intact ceiling images. Secondly, the saliency maps for the damaged ceilings do reflect most of the damage zones which are very hard for traditional algorithms to detect.

3. The images in the test dataset are randomly chosen from the ceiling image collection by the Kawaguchi Lab, within which there are implied similarities among the images, like the building types, the proportion of ceiling area to the whole image area and the photographer's habits, etc. All of these will strongly affect the representations in the ceiling images. Ceiling images from outsider are necessary to validate the saliency map method.

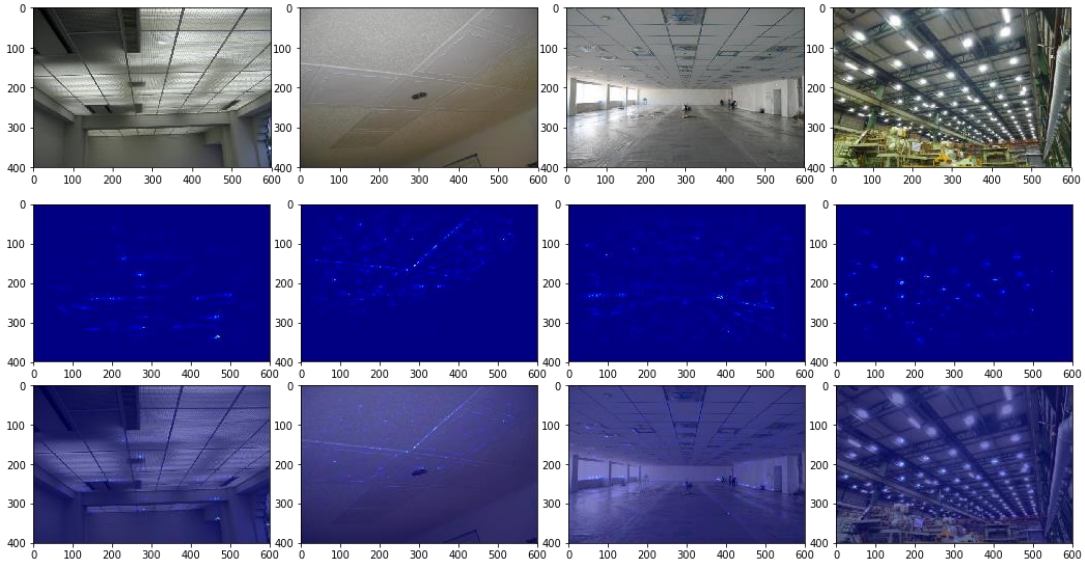




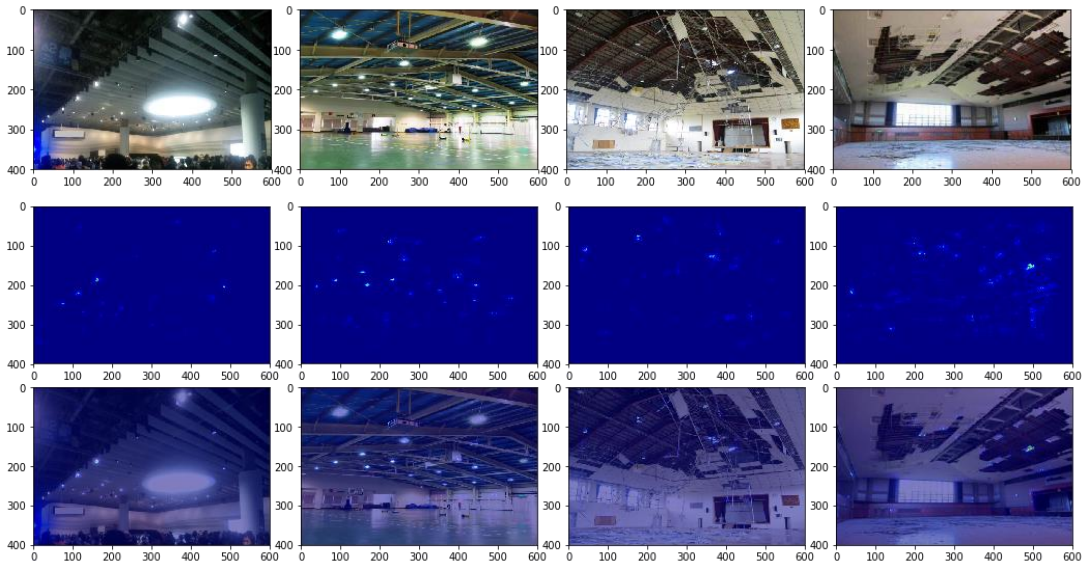
(9) P: 0.059 (10) P: 0.951 (11) P: 0.245 (12) P: 0.590



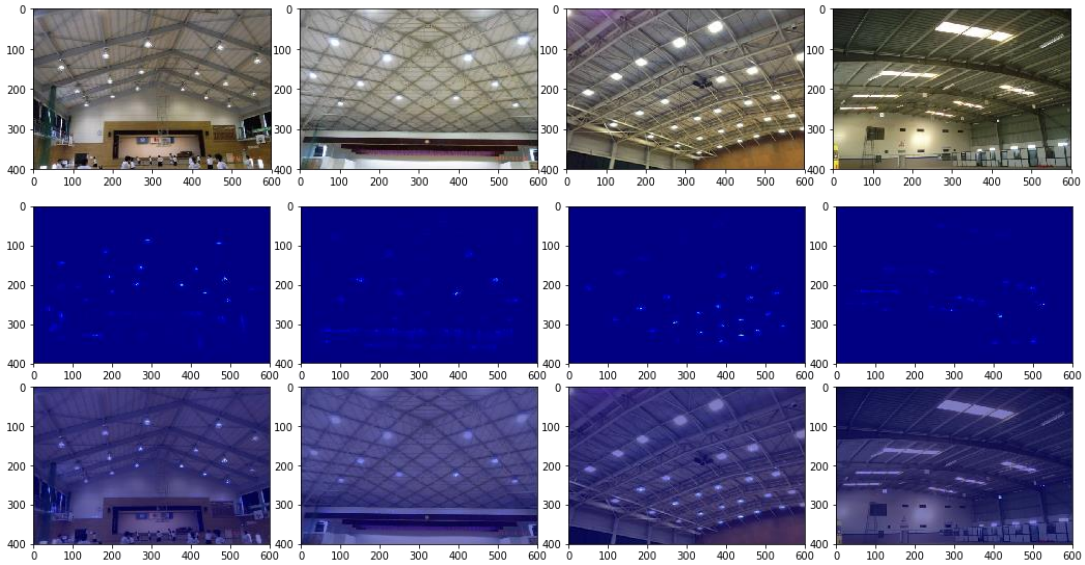
(13) P: 0.409 (14) P: 0.628 (15) P: 0.574 (16) P: 0.002



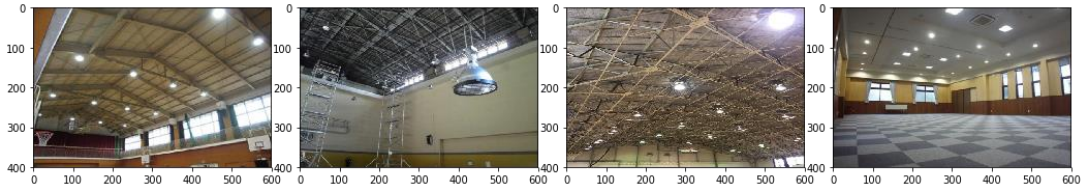
(17) P: 0.316 (18) P: 0.002 (19) P: 0.597 (20) P: 0.747

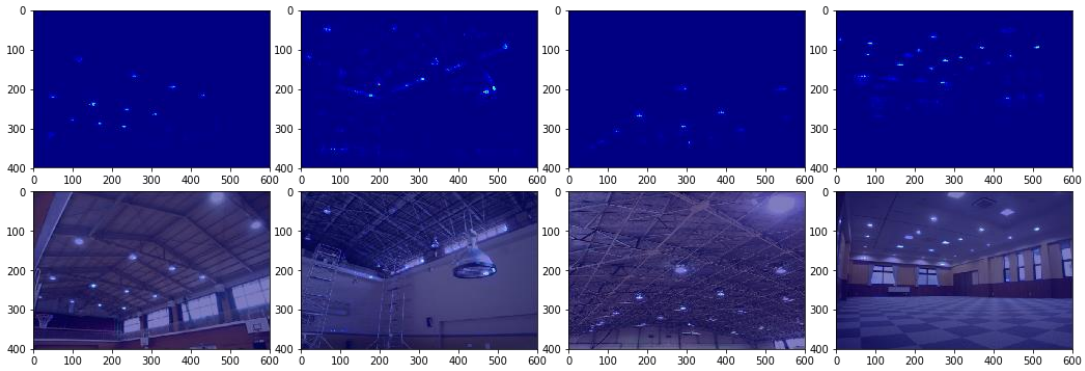


(21) P: 0.068 (22) P: 0.174 (23) P: 0 (24) P: 0

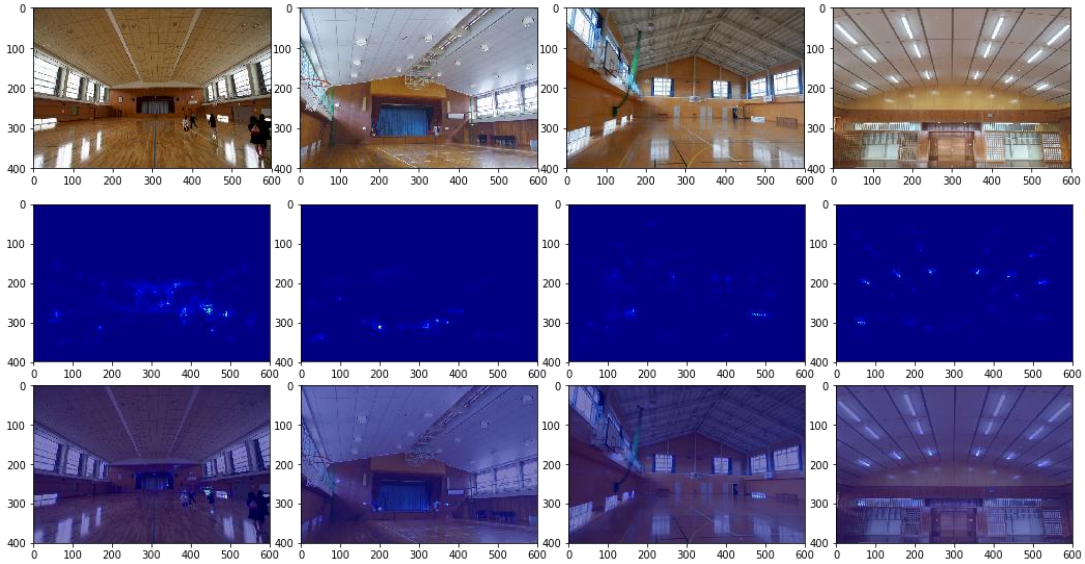


(25) P: 0 (26) P: 0.338 (27) P: 0.032 (28) P: 0.088

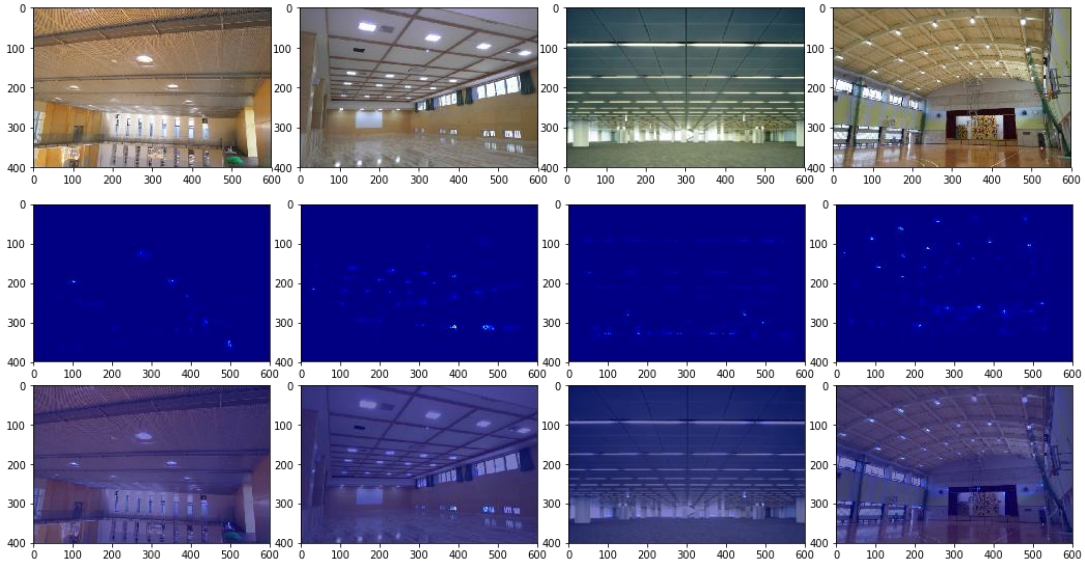




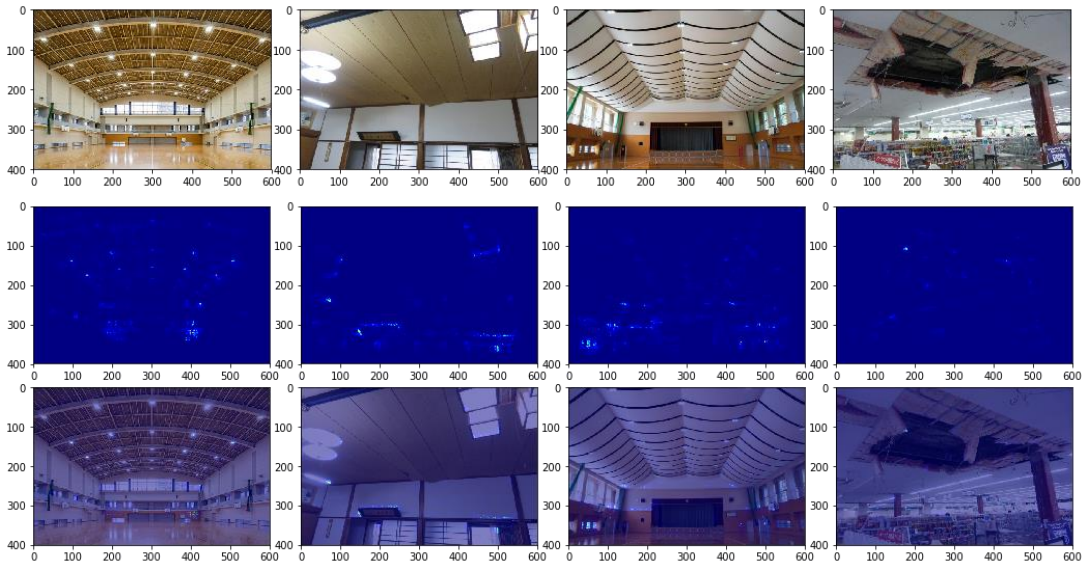
(29) P: 0.087 (30) P: 0.359 (31) P: 0.371 (32) P: 0.207



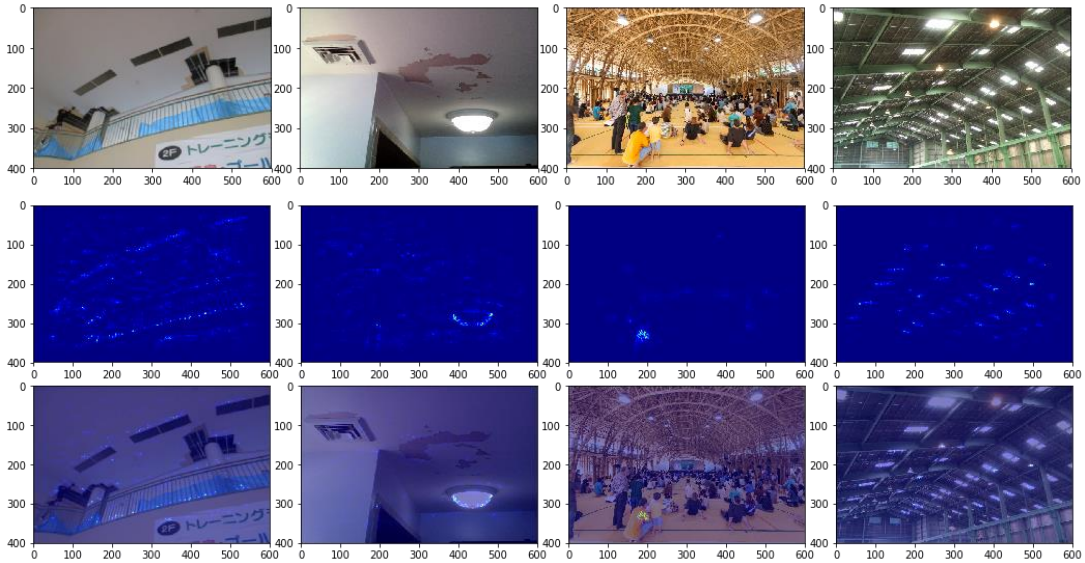
(33) P: 0.232 (34) P: 0.067 (35) P: 0.010 (36) P: 0.020



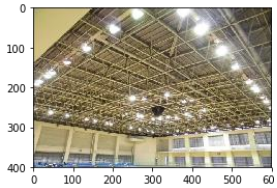
(37) P: 0.009 (38) P: 0.275 (39) P: 0.246 (40) P: 0.691



(41) P: 0.620 (42) P: 0.563 (43) P: 0.038 (44) P: 0



(45) P: 0.001



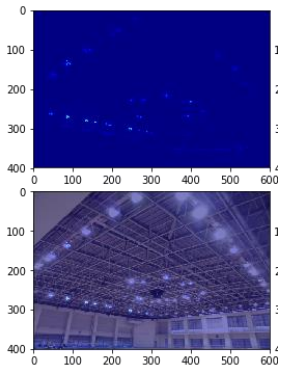


Fig. 4.6 Saliency maps to ceiling images from internet (Fig. 3.24)

The saliency maps in Fig. 4.6 are interpreted as follows:

1. For those correctly predicted intact ceiling images, the most activated pixels are still the well-ordered lights. The crowds are also noticed in them.

2. For those correctly predicted damaged ceiling images, the saliency maps to them can reflect most the damaged parts in the ceilings. However, there are still exceptions that even if the image is correctly predicted, the saliency map to it fails to detect the damage such as the stains and peeling off in the ceilings. That is because there are too few images of such damage forms in the training data (only three images or less are about such damages). But the correct predictions still provide warnings to further notice the ceilings.

3. The incorrectly predicted damaged images are dangerous in practice, because the damages are ignored and evaluated as safe. Some of the damages are voids of ceiling boards with neat cut edges, which are easily confused with vents by the CNN model. Some of the damages are of too much proportion to the whole image that the CNN model would confuse with crowds. There are also damage shapes that have never been shown in the training data within which the CNN model fails to learn.

4. There is only one incorrectly predicted intact image with spotty ceilings. But the saliency map to it also fails to focus on the spots in the ceilings but on irrelevant objects. But notice the prediction to it is 0.502, the noticed objects have low weights to the final prediction.

The saliency maps to the ceiling images from internet prove that they are possible solutions to highlighting the damages in the ceilings with the corresponding to the predictions to them. By highlighting the damages, it also performs damage detection task.

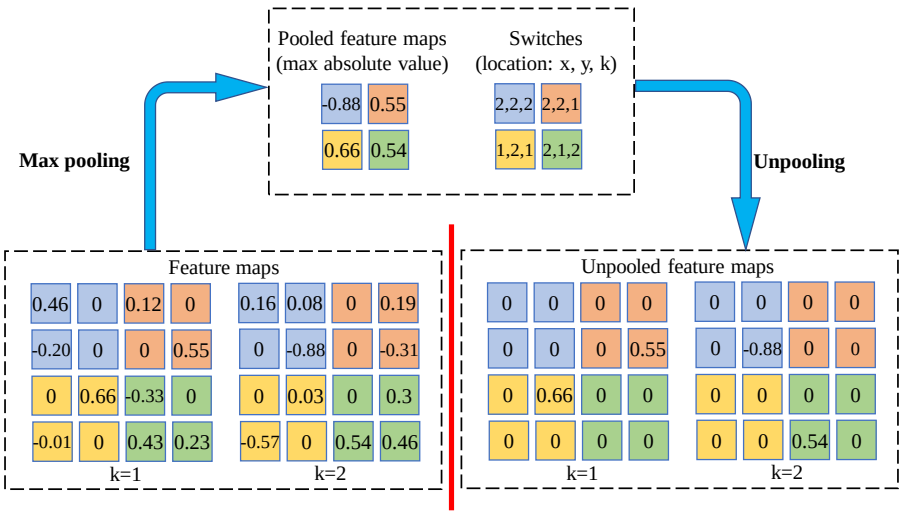
4.2.2 Gradient-weighted class activation mapping (Grad-CAM)

The saliency map method visualizes the attention that the CNN model pays to an image when the image is shown to it, for the first time in convolutional neural networks. Visualization the attention map of a CNN model has at least three merits: (1) It visualize if the CNN model has really learnt corresponding features from the training data; (2) If the CNN model has learnt enough, the attention map can be used as object detection without ever having been explicitly taught by the training data; (3) It connects the research on computer algorithms with the research on human minds. Since then, there are more researches on the attention maps made by the CNN model. The most recent winners in the classification and object detection tasks introduced the attention-aided models to reach state-of-art performances [135, 136]. There is also primitive research on the comparison between human attention and VQA (Visual Question Answering) models attention [137].

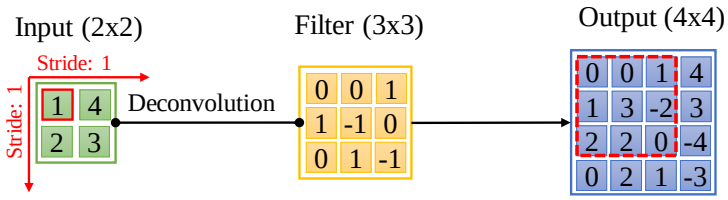
For the ceiling damage detection task in this thesis, the saliency map method provides a solution to it. However, the saliency map method aggressively simplifies the complex non-liner convolutional neural networks into a linear function (using the first-order Taylor expansion). The results of saliency map are satisfying, but there are still improvements could be done. In this section, a more precise method, Gradient-weighted Class Activation Mapping (Grad-CAM), is introduced to visualize better attention maps. Before that, prerequisite knowledge is:

1. Deconvolutional networks

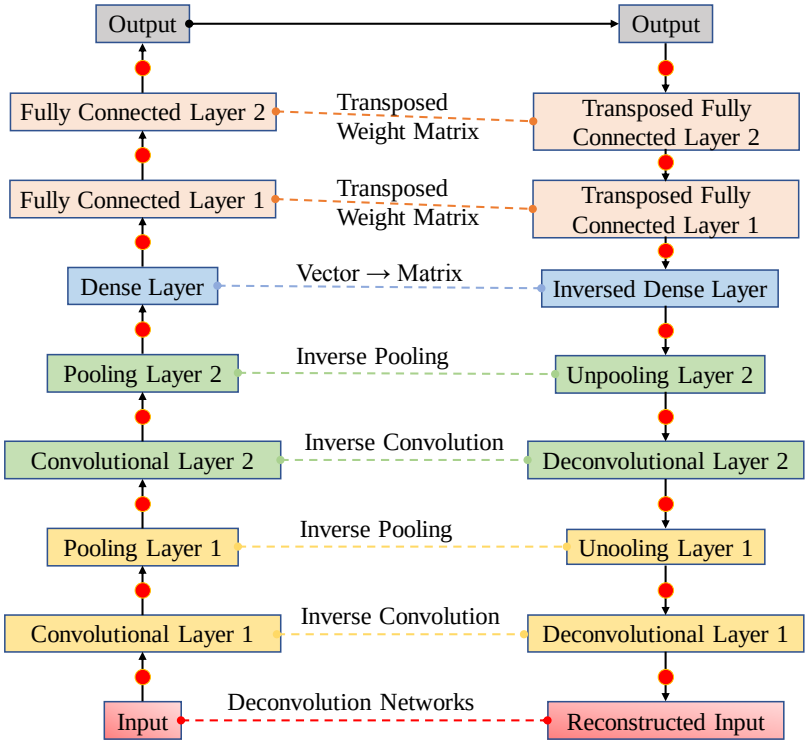
Researchers are trying to open the black box of deep learning from different aspects, the deconvolutional perspective is one of them [120, 125]. Contrary to the down-sampling convolutional operation, the deconvolutional layers performs up-sampling operation. To a deconvolutional networks model, the output is a reconstructed image the same size to the input image, emphasizing the pixels that most activate the neurons in the convolutional layers. Generally, the deconvolutional networks are inverse neural networks to the convolutional neural networks. To realize the inverse operations to a CNN model, the unpooling operation and deconvolutional (also named transposed convolutional) operation are used to aid the output of the final reconstructed image (shown in Fig. 4.7). The purpose of deconvolutional networks is to visualize the most activated pixels in a trained CNN model by the reconstructed input.



(a) Unpooling



(b) Deconvolutional operation [116]

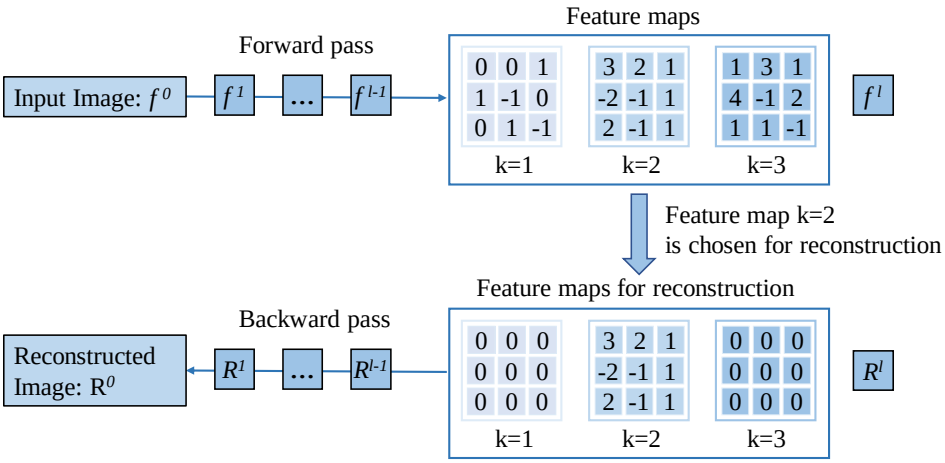


(c) From input to reconstructed input

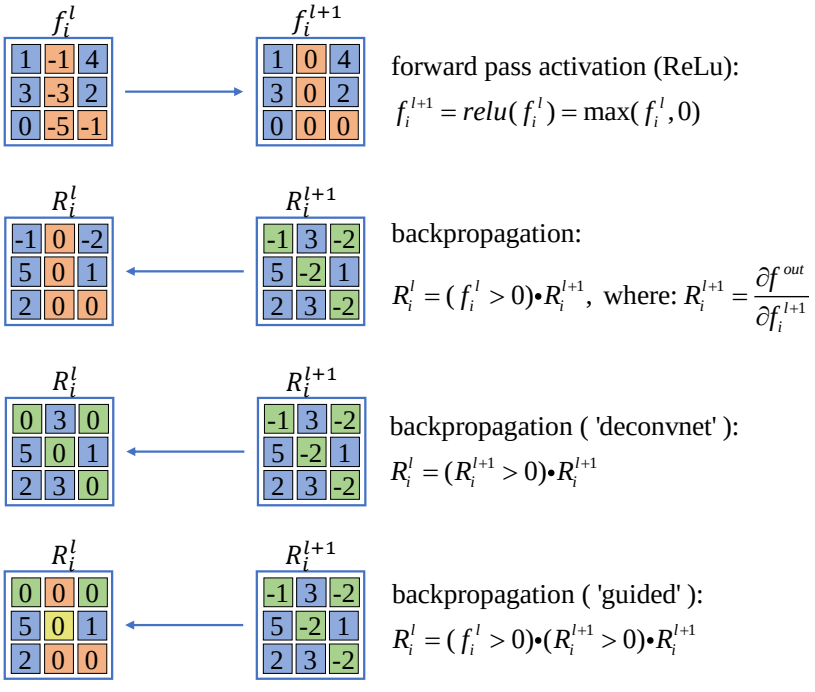
Fig. 4.7 Deconvolutional Networks

The research of deconvolutional networks provides profound understandings of

convolutional neural networks with visualizing the gradually abstraction in the convolutional layers. Fig. 4.7(c) shows the data flow in deconvolutional networks: given an input to a trained CNN model, the details (switches in the pooling layers and deconvolutional matrices) of gradually abstractions in the forward propagation process are recorded to provide traces in the deconvolutional process (red dots represent data node). More generalized schematic visualizations to the deconvolutional networks and details are shown in Fig. 4.8 (corresponding data nodes are picked from forward pass and backpropagation [138]).



(a) Given an input image, reconstruct the image for layer l



(b) Different methods of backpropagation

Fig. 4.8 Details of Deconvolutional Networks

2. Class activation mapping (CAM)

In 2015, a new architecture of CNN named “the all convolutional networks” that consists solely of convolutional layers yields competitive or state-of-art performance on several object recognition datasets [138]. It uses the global averaging pooling layer (GAP) as the penultimate layer to perform more extreme dimensionality reduction than the max pooling layer (the last layer is softmax layer for object categorization). The global averaging pooling layer reduces a three-dimensional matrix ($h \times w \times d$) into a tensor ($1 \times 1 \times d$) [139] (shown in Fig. 4.9).

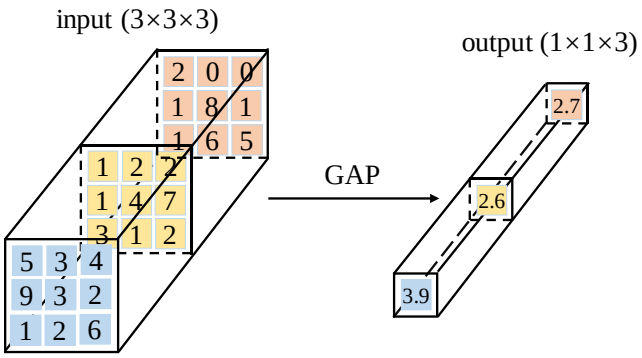


Fig. 4.9 Global averaging pooling (GAP)

In mid-2016, a new architecture of CNN named Class Activation Mapping (CAM) using the all convolutional neural networks with the GAP layer as classifier is demonstrated that such architecture of CNN model can be used for not only object classification but also for object recognition at the same time [140]. This means that a CNN model with the CAM architecture is able to tell what an object in an image is and where it is within only one forward pass. The main idea of CAM is that each class activation map compressed by the GAP layer acts as an object detector and an object classifier at the same time. Fig. 4.10 shows how the CAM architecture performs object classification and object detection function.

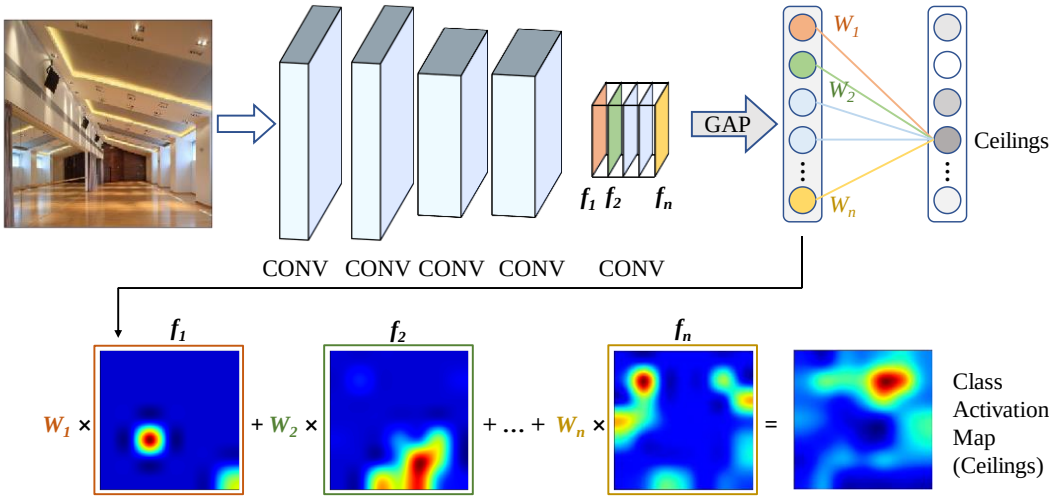


Fig. 4.10 Class Activation Mapping (CAM)

The last layer shown in Fig. 4.10 is possible to be modified as a softmax layer, a regression layer or other kinds of losses. In the case of a softmax layer as the final layer, the description to a CAM model is as:

For a trained CAM model, an input image is down-sampled gradually when transferring through convolutional layers. For the last convolutional layer, f_k represents the *k-th* feature map with n feature maps in total. f_k is a two-dimensional matrix with spatial information of the input image. In fact, f_k is the visualization map of the pattern k. After f_k is fed into the GAP layer, the result is F^k, where $F_k = \frac{\Sigma f_k^i}{Z}$ (Z is the total number of digits in f_k). For a given class c, the score $S_c = \Sigma_k \omega_k^c F_k$, where ω_k^c is the weight of class c for the *k-th* feature map. The probability of the image of being class c P_c is calculated through the softmax function: $P_c = \frac{\exp(S_c)}{\Sigma_c \exp(S_c)}$.

Recall the score for class c:

$$S_c = \sum_k \omega_k^c F_k = \sum_k \omega_k^c \frac{\Sigma f_k^i}{Z} = \sum \sum_k \omega_k^c \frac{f_k^i}{Z} \tag{4.11}$$

Define M_c as the class activation map for class c, where:

$$M_c = \sum_k \omega_k^c \frac{f_k^i}{Z} \qquad 4.12$$

Then, $S_c = \sum M_c$. M_c indicates the importance of the activation as the image classified to class *c*.

This technique named Class Activation Mapping (CAM) enables a CNN model trained for classification objective can be directly used for object localization tasks. The idea behind CAM also aids other researchers to understand the convolutional neural networks in their practices.

3. Gradient-weighted class activation mapping (Grad-CAM)

The Class Activation Mapping (CAM) method allows a all-convolutional model perform object localization even if it is not purposely trained for object localization. The biggest drawback of CAM is that it restricts the architecture of the CNN model, which only containing convolutional layers and a global average pooling before the last softmax prediction layer. In a few months after the CAM method announced, a more generalized method named Gradient-weighted Class Activation Mapping (Grad-CAM) , using the gradients of any target class flowing into the last convolutional layer to perform localization is issued [141]. The Grad-CAM method is shown in Fig. 4.11. Given an image and a category ('ceilings') as input, the objective is to locate the ceiling zone in the image. Firstly, the image is forward propagated through the trained CNN model until it reaches the raw class scores before the softmax layer. Secondly, the vector representing the desired class ('ceilings' in this example), with only the desired class as 1 and all other classes as 0, is backpropagated till to the last convolutional-shaped layer (the last pooling layer in this example). Finally, a Grad-CAM localization heatmap representing the desired class is generated by combining the weights from the pooling layer and the vector backpropagated to the pooling layer.

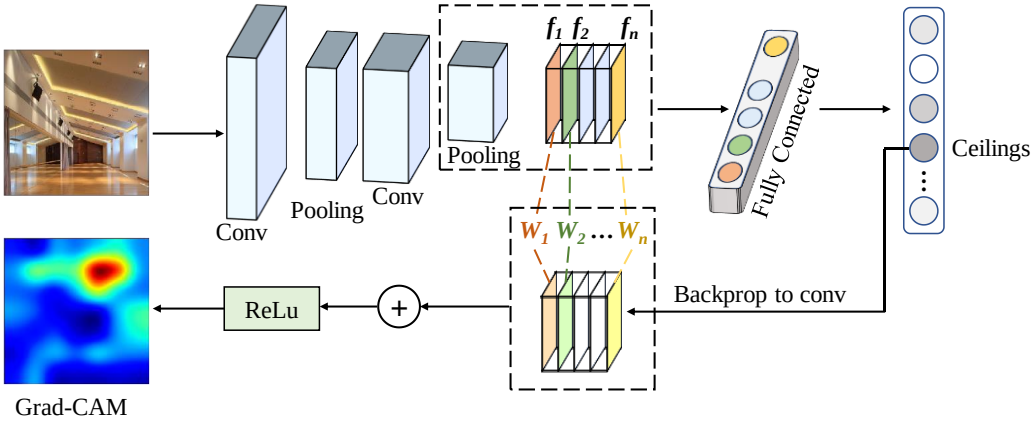


Fig. 4.11 Grad-CAM

As shown in Fig. 4.11, the objective is to obtain the localization map to the given image for the desired class *c*. The final output is an image $L^c_{Grad-CAM} \in \mathbb{R}^{w \times h \times d}$ with width *w*, height *h* and depth *d* (usually 3) for class *c*. Firstly, compute the gradient matrix $g^c_k = \frac{\partial S_c}{\partial f_k}$,where S_c is the score for class *c* before propagated to softmax, f_k is the *k-th* feature map in the last convolutional-shaped layer. Then the neuron importance weights W^c_k is computed by flown through a global average pooling operator (*Z* is the total number of digits in g^c_k):

$$W^c_k = \overbrace{\frac{1}{Z}\Sigma}^{\text{global average pooling}} \underbrace{g^c_{k,i}}_{\text{gradients by backpropagation}} = \frac{1}{Z}\Sigma\frac{\partial S_c}{\partial f^i_k} \tag{4.13}$$

W^c_k represents the weights of the *k-th* feature map f_k going downstream through the deep networks in which the layers after the last convolutional-shaped layer (the last pooling layer in this example) for the representation of the targeted class *c*. The final output of $L^c_{Grad-CAM} \in \mathbb{R}^{w \times h \times d}$ is calculated as:

$$L^c_{Grad-CAM} = \mathrm{Re}\,Lu(\sum_k W^c_k f_k) \tag{4.14}$$

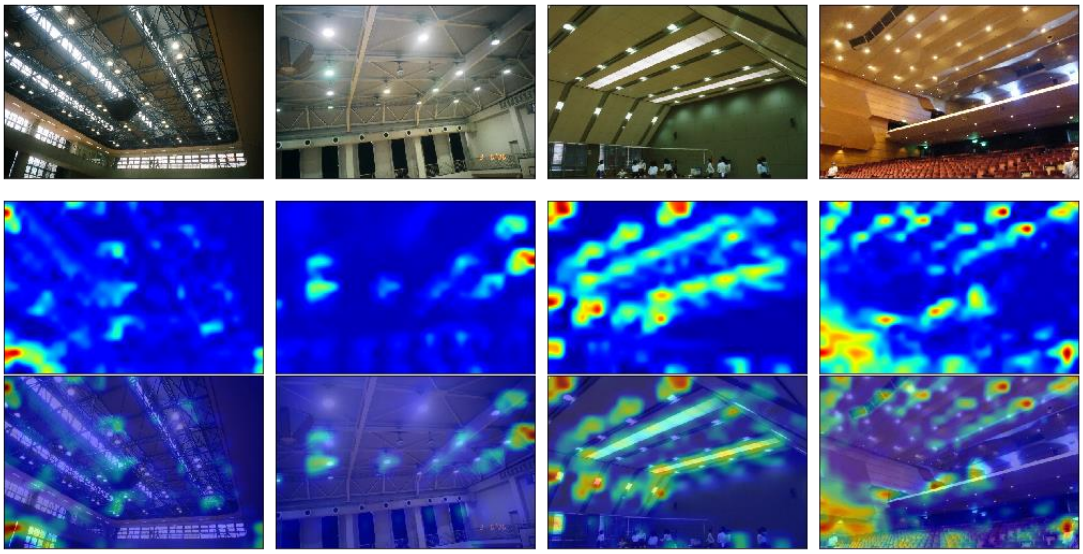
From Fig. 4.10 and Fig. 4.11, we can find that the weights in the Grad-CAM method are generated by the gradients in backpropagation process, through which the applicable

range of Grad-CAM is widened into deep learning networks with convolutional layers, which is also applicable to the CNN model we trained for ceiling damage evaluation.

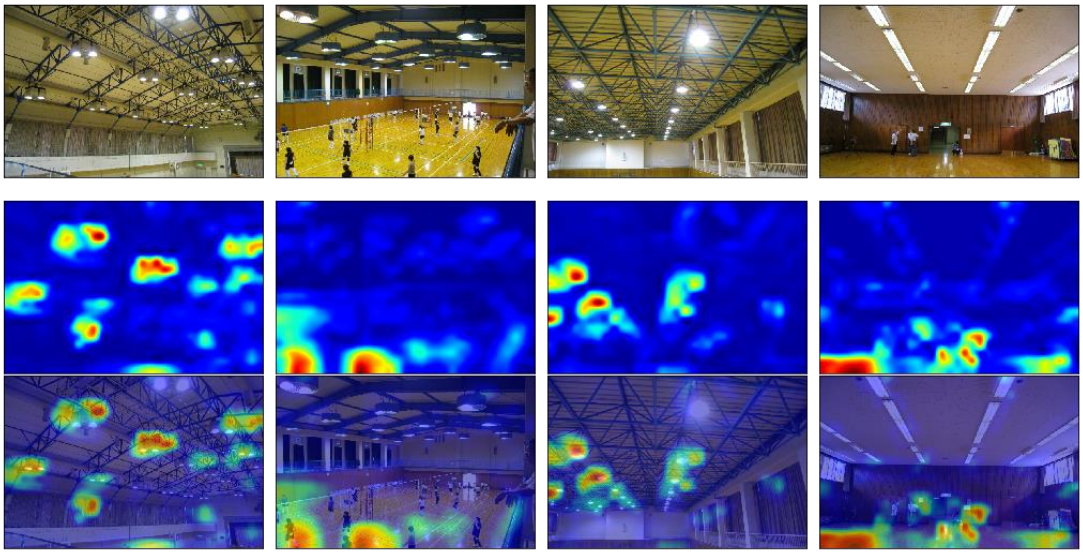
4. Applying the Grad-CAM method in the CNN model for ceiling damage evaluation

As Fig. 3.8 shows, the last convolutional-shaped layer in the CNN model is the Pooling Layer 15. The CNN model output only one node representing the probability of the ceilings in the image are damaged. Based on these and the Grad-CAM method, the same images to Fig. 4.5 and Fig. 4.6 are chosen for damage region localization.

(1) P: 0.019 (2) P: 0.018 (3) P: 0 (4) P: 0.266



(5) P: 0.002 (6) P: 0 (7) P: 0.001 (8) P: 0.010



(9) P: 0.747 (10) P: 0.850 (11) P: 0.673 (12) P: 0.915

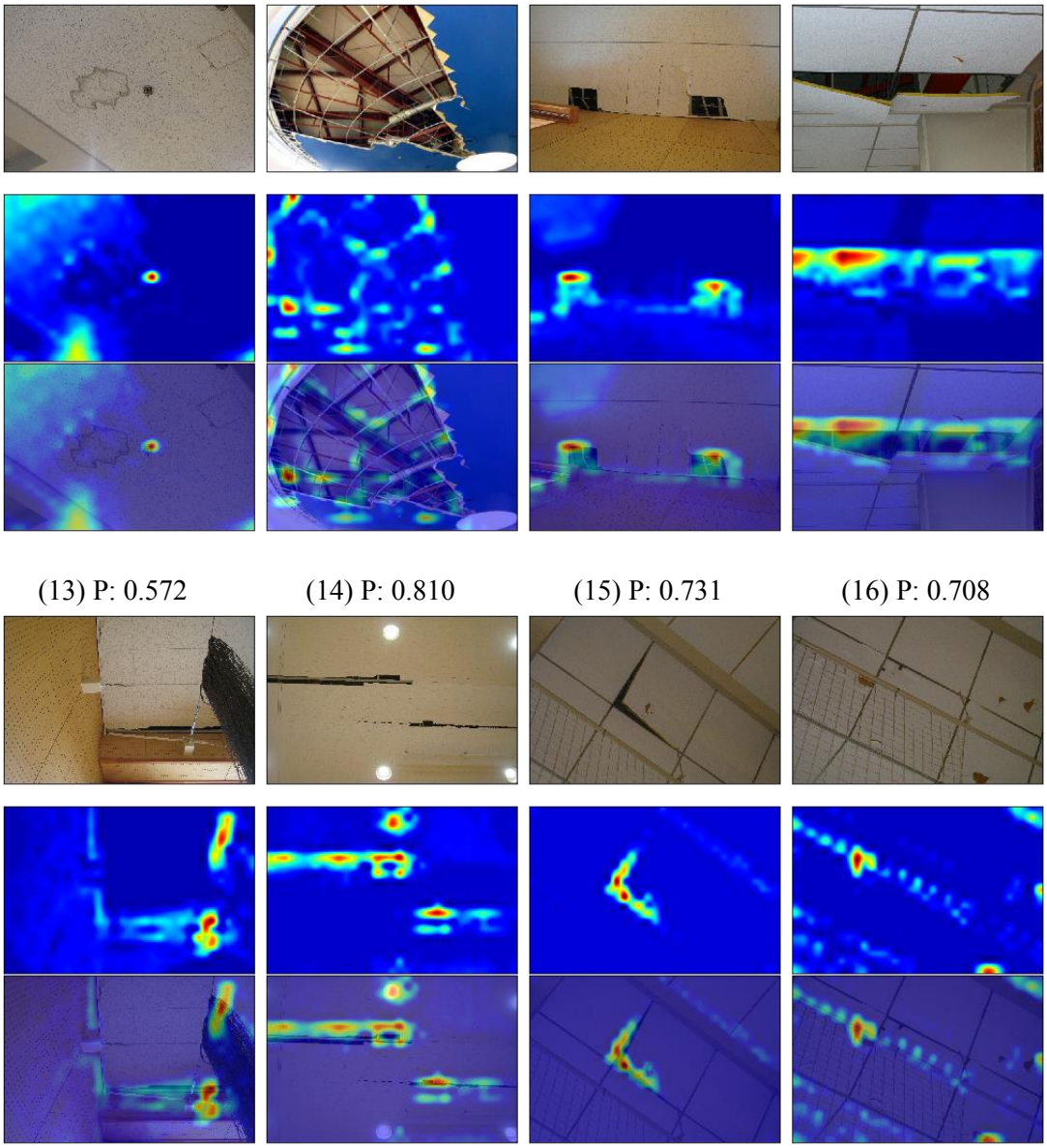
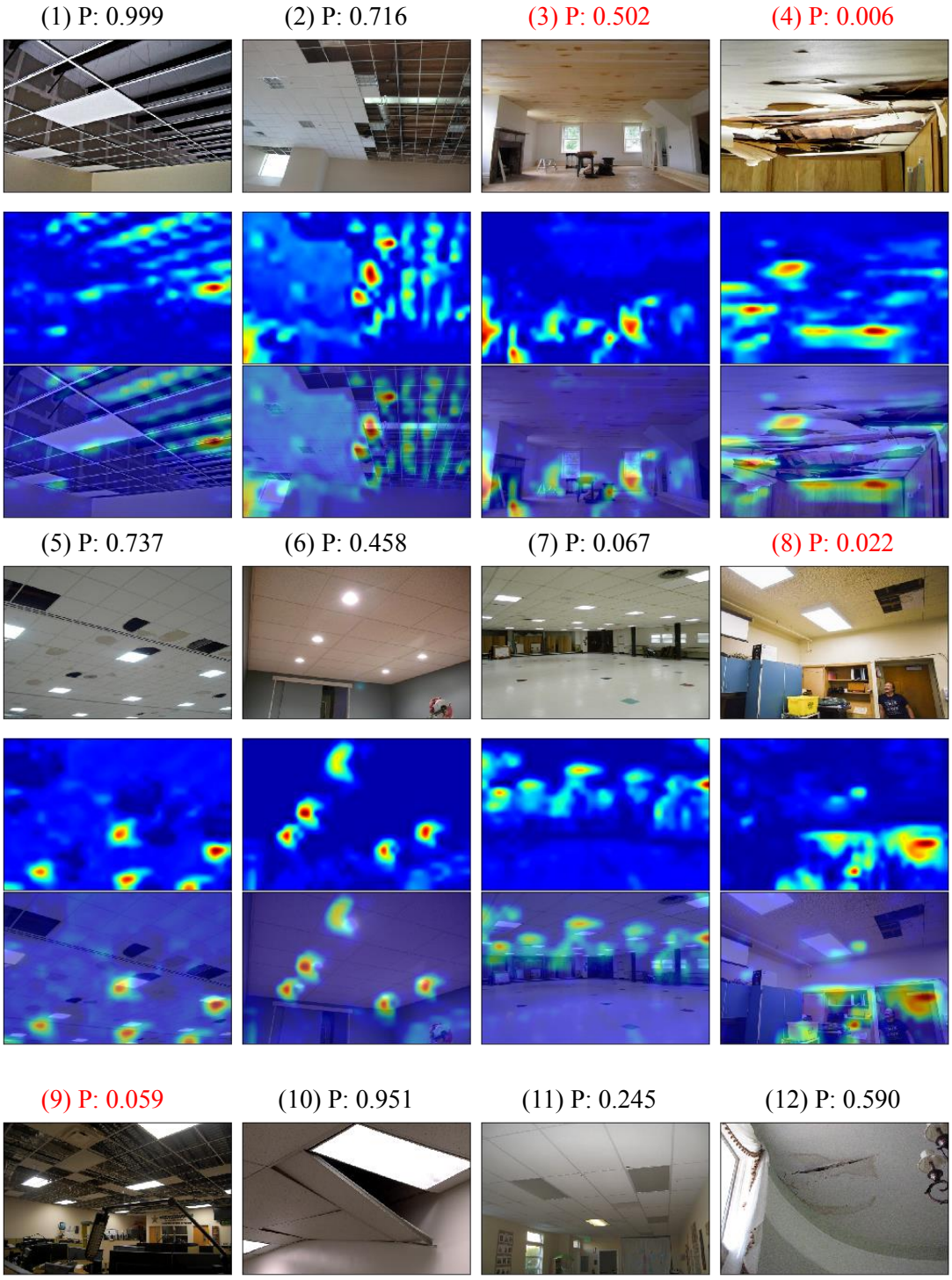
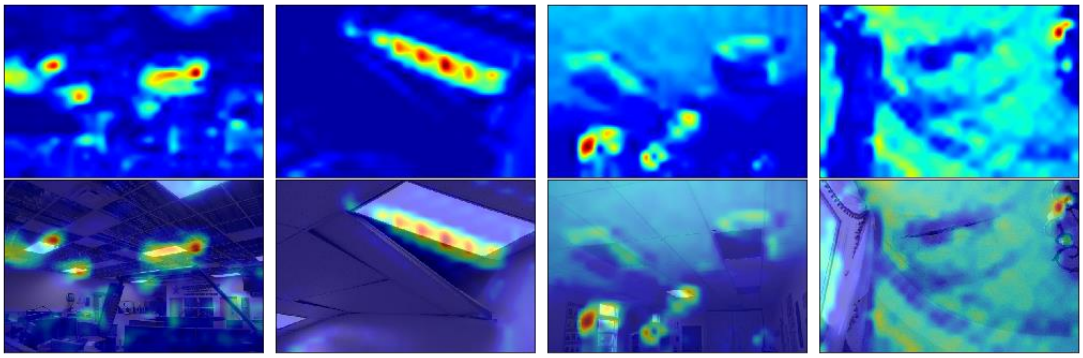


Fig. 4.12 Grad-CAM to some ceiling images from the test dataset

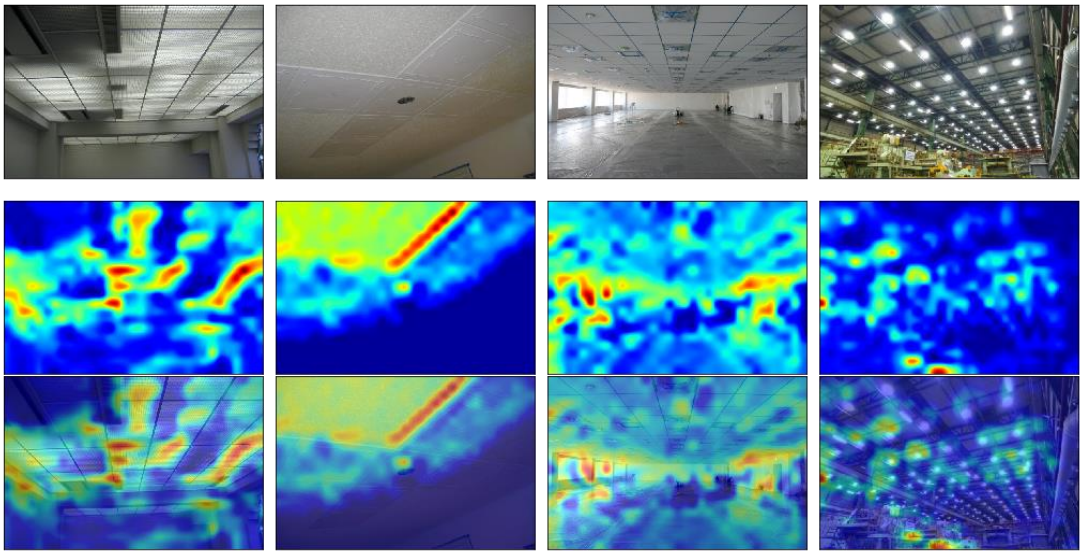
In Fig. 4.12, both intact ceiling images and damaged ones from test dataset are chosen for Grad-CAM method to perform damage region detection. For the intact ceilings, the final predictions are very small in digits and the most activated regions are lights in the ceilings for most of the images. While Grad-CAM of images Fig. 4.12 (6) and (8) focus on objects irrelevant to ceilings like floors and people, this reveals that objects like floors and people share low weights to the final predictions. This consists with reality that floors and people in an image indicate that the structural space is probably safe. For the latter damaged ceiling images in Fig. 4.12, the overall damage detection performed by Grad-CAM is very satisfying. The attention maps not only detect the damage locations but also draw the contours of the damages, which providing strong guidance to the user. Except for Fig. 4.12(9), the Grad-CAM fails to detect the obvious water stains on the ceiling board. The reason for this is already discussed in the saliency

map section: the lack of water stain images in the training data fails to teach the CNN model to learn such damage form.

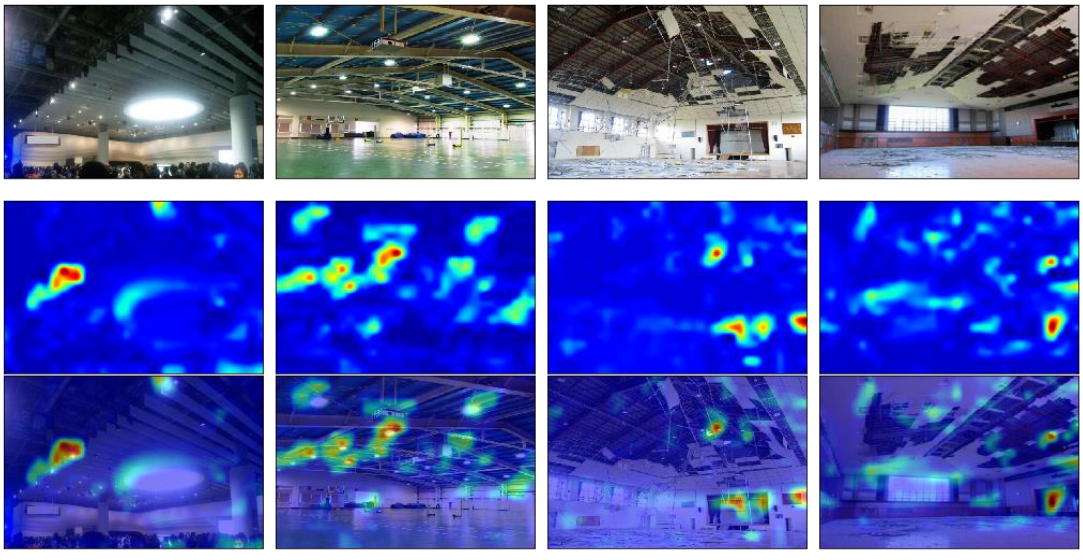




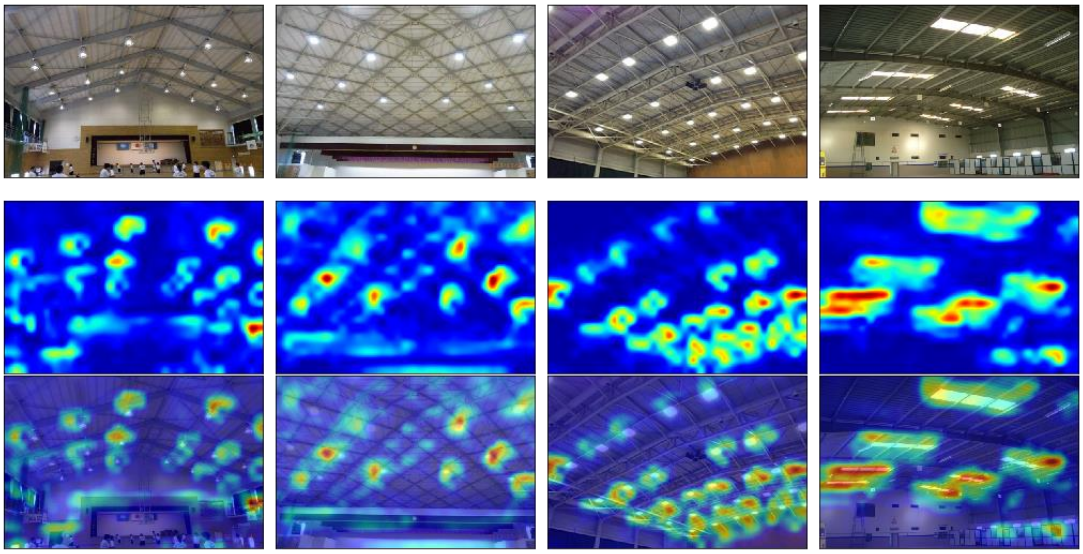
(13) P: 0.409 (14) P: 0.628 (15) P: 0.574 (16) P: 0.002



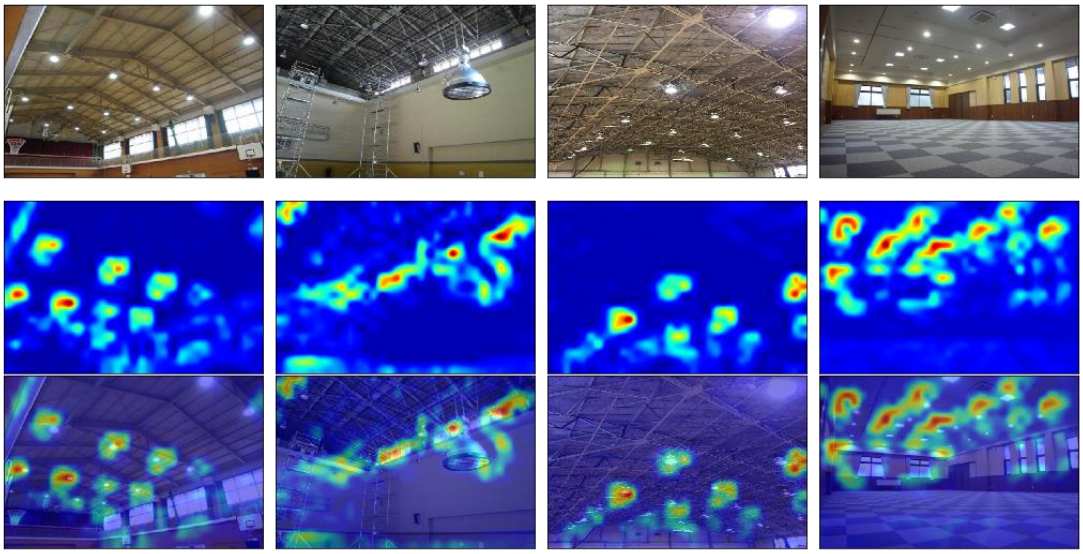
(17) P: 0.316 (18) P: 0.002 (19) P: 0.597 (20) P: 0.747



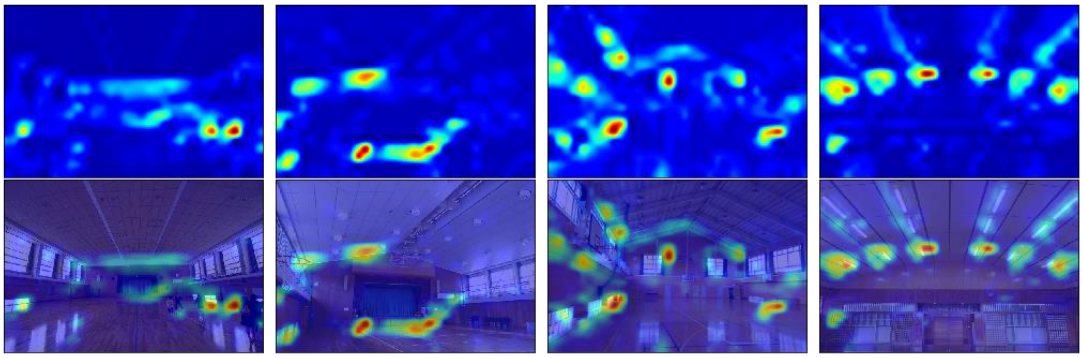
(21) P: 0.068 (22) P: 0.174 (23) P: 0 (24) P: 0



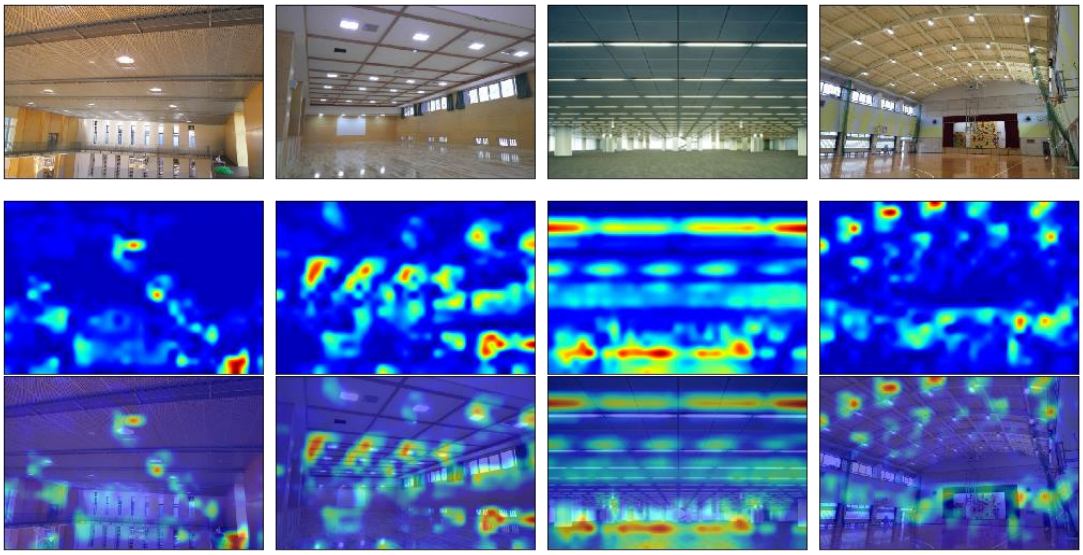
(25) P: 0 (26) P: 0.338 (27) P: 0.032 (28) P: 0.088



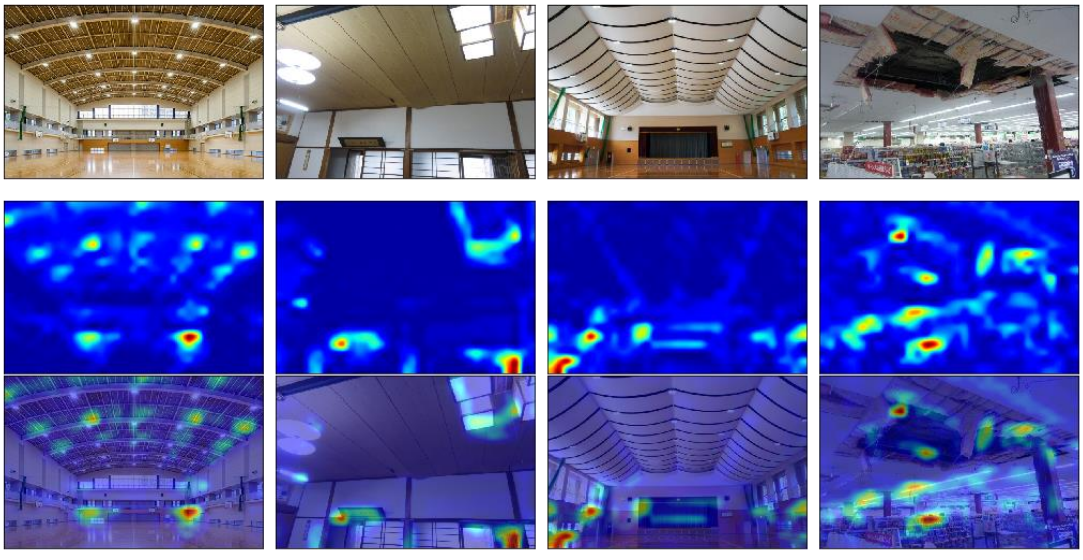
(29) P: 0.087 (30) P: 0.359 (31) P: 0.371 (32) P: 0.207



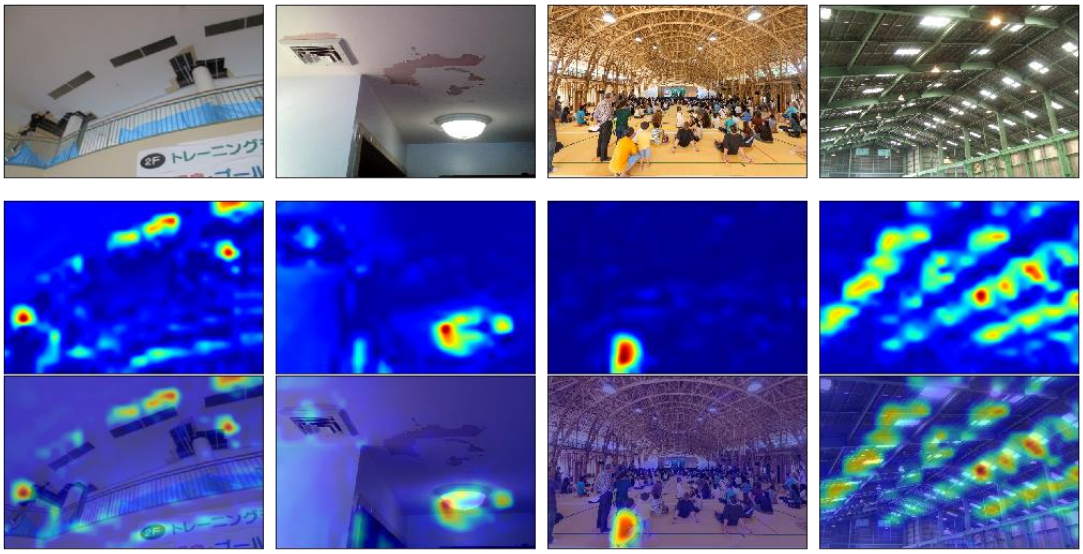
(33) P: 0.232 (34) P: 0.067 (35) P: 0.010 (36) P: 0.020



(37) P: 0.009 (38) P: 0.275 (39) P: 0.246 (40) P: 0.691



(41) P: 0.620 (42) P: 0.563 (43) P: 0.038 (44) P: 0



(45) P: 0.001

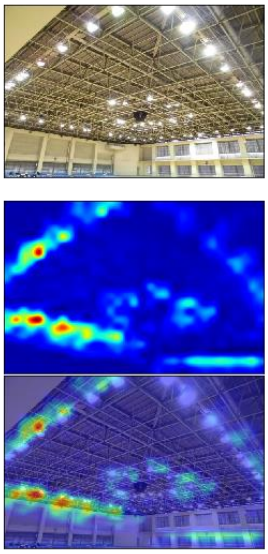


Fig. 4.13 Grad-CAM to ceiling images from internet

For the ceiling images from internet (Fig. 4.13), the Grad-CAM of them are discussed into four categories:

1. For those correctly predicted intact ceiling images, the digital predictions to them are below 0.5. The most activated regions in the Grad-CAM are bright regions like lights and windows. Noticing that the bright regions are usually regularly distributed in the image, which indicating the ceilings behind them are intact in high probability. This could be a characteristic for the CNN model to predict that the ceilings are intact.

2. For those correctly predicted damaged ceiling images, the Grad-CAM to them are not as precise as those to the test dataset. However, they still provide instructive guidelines to the user. The lights in the images may distract the CNN model to some extent, damages in the ceilings are still noticeable. However, in Fig. 4.13(12) the stains in the ceiling seems to be deliberately ignored, which is also a kind of damage region detection in a negative way.

3. For those incorrectly predicted damaged images (Fig. 4.13(4), (8), (9), (38)), the Grad-MAP to them can be interpreted in different ways. Fig. 4.13(4) has the incorrect prediction but correct region activations. While Fig. 4.13(8) is completely incorrect both in prediction and region activations. It fails to notice the absence of the ceiling boards or the lights. One possible explanation is that there are no, not even very few images in the training data like such interior like an office from the perspective in Fig. 4.13(8), which fails the CNN model to learn such circumstance. Damages in Fig. 4.13(9) are easily to be confused with regularized decorations at the first glance even for human. The Grad-CAM to it also indicates that the CNN model mistakes it with an intact ceiling image because it focuses on the lights in the ceilings. In Fig. 4.13(38), the CNN model notices very slightly on the tilted edges in the ceiling board shown by the Grad-CAM.

4. Fig. 4.13(3) is the only incorrectly predicted intact image with the prediction 0.502. From the Gram-CAM to it, we can find that the model dose not notice the ceilings but notice other objects like the windows and the floors.

From the comparison between the attention maps of the Saliency Map and the Grad-CAM, it can be confirmed that both of them reflect the attention paid by the CNN model when given an image. The Saliency Map method uses more aggressive approximation that omits many details in the attention map. However, sometimes the Saliency Map method abandons irrelevant distractions and helps the user to stay focused on the most important regions. To the contrary, the Grad-CAM method uses more exquisite approximation to keep many details from the final predictions mapping back to the input layer. The Grad-CAM method provides rich details of the attention paid by the CNN model, sometimes much too rich. A good solution is to combine the merits of these two methods and avoid the defects of them.

4.3 Building a ceiling damage detection system

4.3.1 The workflow of a ceiling damage detection system

Based on the previous work for training and interpreting the CNN model, the flow chart

of the ceiling damage detection system using convolutional neural networks is shown in Fig. 4.14. The system performs attention evokes to the user by highlighting the damaged regions. The user (an expert in ceiling structures or an amateur) is involved in the decision-making process. This system is possible to improve the prediction accuracy even if the CNN model makes incorrect predictions because the user is involved in the damage detection process. The workflow is:

1. Start.

2. The user takes a ceiling image to perform ceiling damage detection. The image is usually in high resolution (for example, an iPhone X has the 12MP cameras that can take photos with the resolution of 3024×4032×3).

3.The image is resized into the resolution of 400×600×3 by the system.

4. Send the resized image into the trained CNN model.

5. Three kinds of outputs are generated by the CNN model: a digital prediction (ranging from 0 to 1), a saliency map and a Grad-CAM attention map. The digital prediction is the basic evaluation to the image. The saliency map and the Grad-CAM attention map are two kinds of heatmap that can alert the user with regions that are possibly damaged.

6. The user will decide if it needs more investigations and details to the highlighted regions.

7. If the ceilings need more investigations, the user has two options: a. Crop the regions in the original high-resolution image proportional to the desired regions and send them to the CNN model again, or b. Take more new photos to the desired regions and send the new photos to the CNN model for more detailed damage detection.

8. The user judges if the damaged regions have been correctly detected.

9. If the answer to step 8 is yes, output the image with the damaged regions.

10. If the answer to step 8 is no, the CNN model has made false positive / negative predictions.

11. The ceiling image is possible to be saved as future teacher dataset.

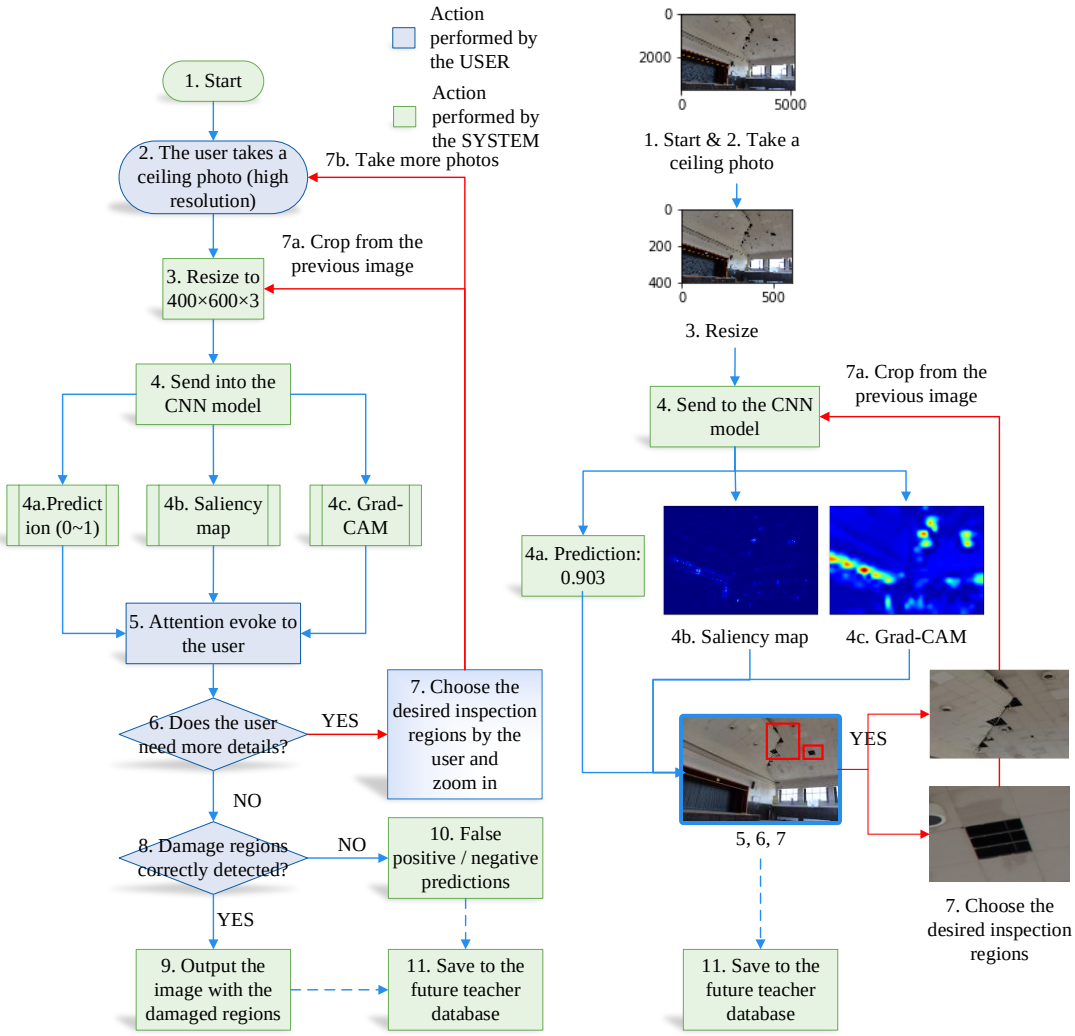


Fig. 4.14 The ceiling damage detection system

Based on the workflow shown in Fig. 4.14, it is possible to build a web-based ceiling damage detection system that can be updated occasionally for multipurpose users (show in Fig. 4.15). This web-based ceiling damage detection system contains mainly two parts: 1. the front-end for users who upload ceiling images, receive the evaluations to the images and make interactive performance with the CNN model (the same flow shown in Fig. 4.14); 2. the back-end for the ceiling damage detection system maintenance personnel who classify, label the new incoming images and re-train the CNN model using new ceiling images added in the training dataset. This system is web-based, meaning it can provide ceiling damage detection service where there is internet connection.

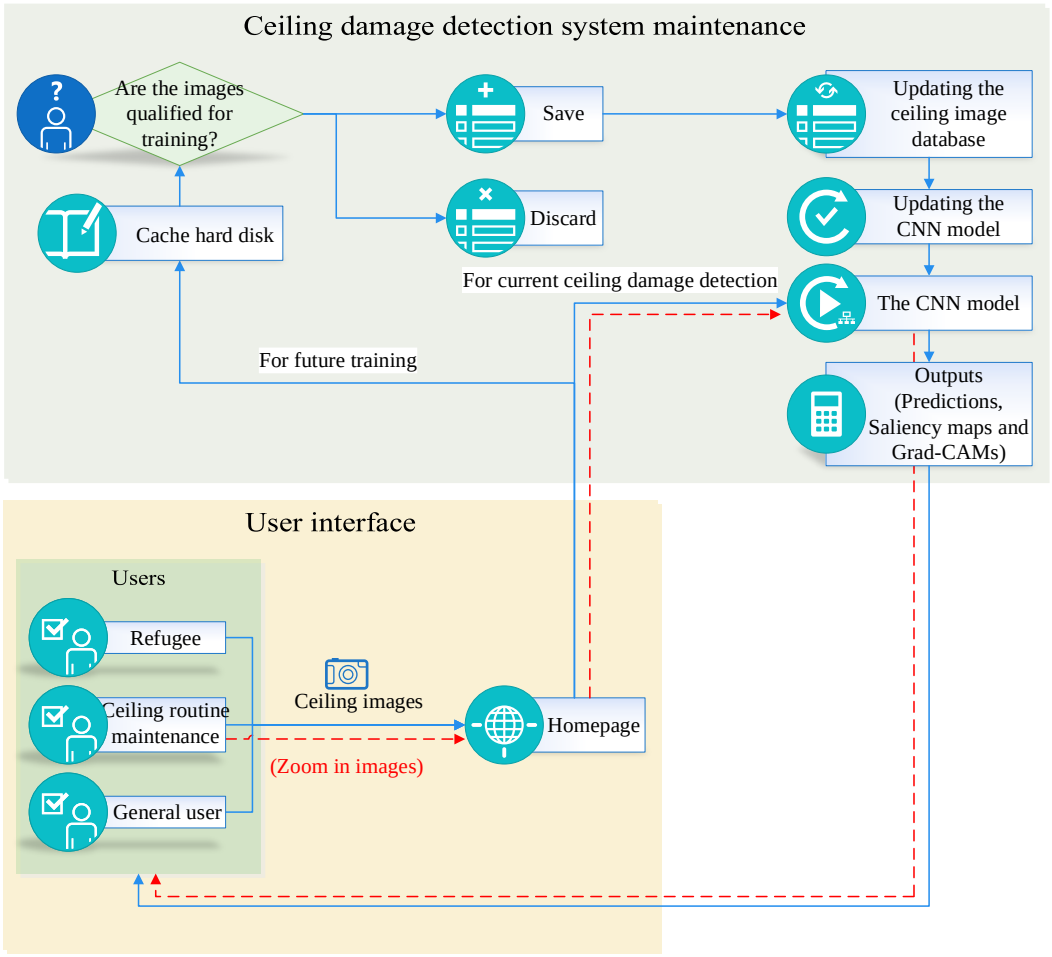


Fig. 4.15 A web-based ceiling damage detection system

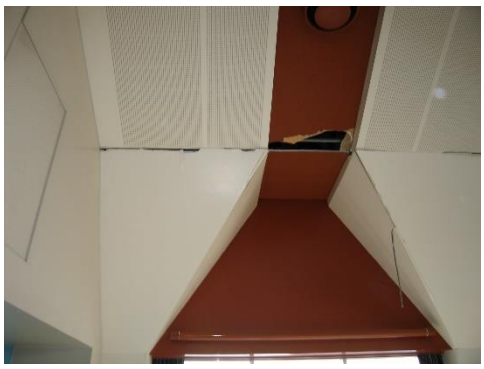
4.3.2 Showcases to the ceiling damage detection system

It is difficult to evaluate the performances of the ceiling damage detection system in Fig. 4.14 by using quantitative indicators such as accuracy or precision because this system involves with the user, who is human. However, it is possible to generate showcases of the interactivities between the user and the ceiling damage detection system to grasp the idea and to investigate the performance of this system.

Three showcases are chosen to display (images shown in Fig. 4.16). The images have never been shown to the CNN model during training.

(a) Showcase 1: damaged ceilings,
resolution: 5184×3456×3

(b-1) Showcase 2: damaged ceilings,
resolution: 2161×1316×3



(b-2) Showcase 2: damaged ceilings from a
nearer perspective,
resolution: 2592×1944×3

(c-1) Showcase 3: intact ceilings,
resolution: 2048×1536×3

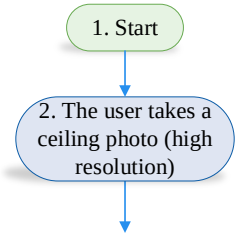
(c-2) Showcase 3: intact ceilings from
another perspective,
resolution: 2048×1536×3

Fig. 4.16 Ceiling images for showcases

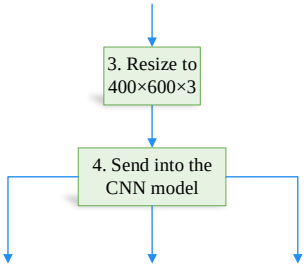
1. Showcase 1(Fig. 4.16(a)):

The process and the information flow in the ceiling damage detection system for the

images in Fig. 4.16(a) are shown below:

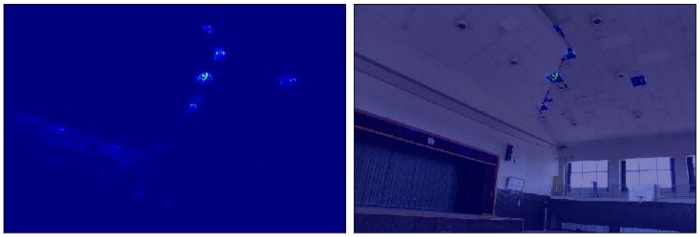


STEP 1&2. Start from the original ceiling image
(resolution: 5184×3456×3)

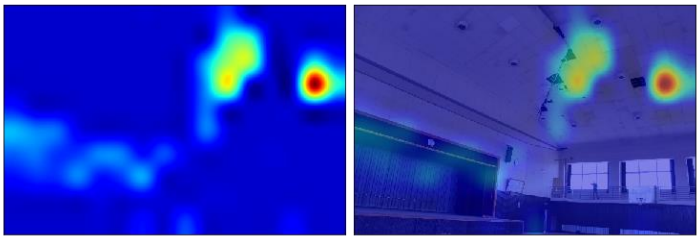


STEP 3. Resize to resolution: 600×400×3 and STEP 4. send to the CNN model

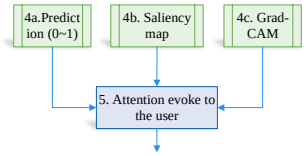
4a. Prediction: 0.903



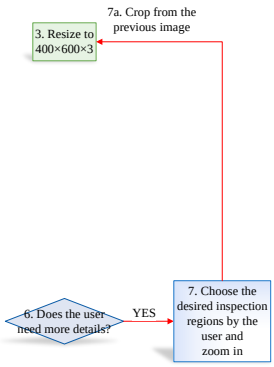
4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image

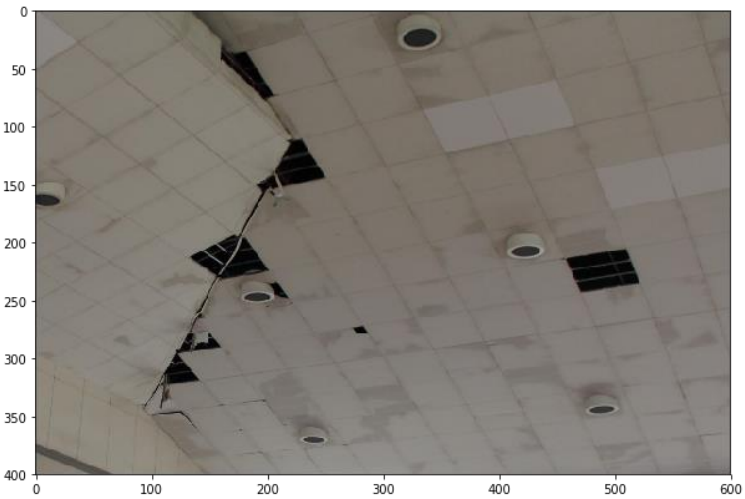


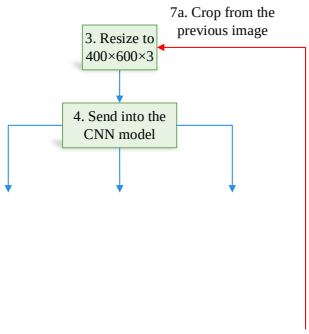
STEP 5. Possible damaged region highlighted to the user



STEP 6. The user makes comprehensive considerations to the outputs and decides to see more details on the upper right region (strongly highlighted) and ignores the weaker highlighted region on the lower left region

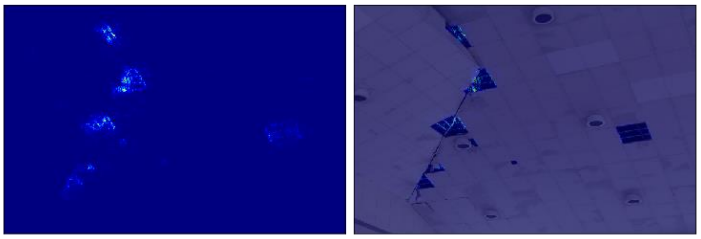
STEP 7a. Crop the highlighted regions by the user from the original image (resolution: 2345×1790×3)



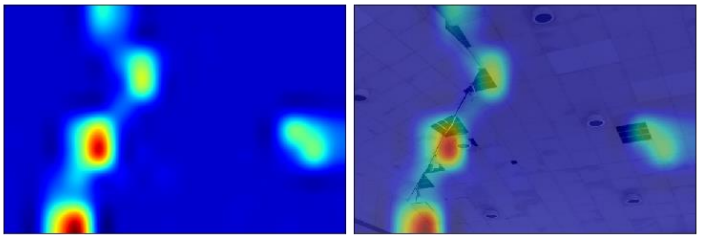


NEW STEP 3. Resize the cropped image to resolution: 600×400×3 and
NEW STEP 4. Send to the CNN model again

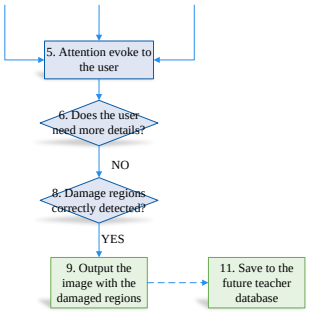
4a. Prediction: 0.963



4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image



NEW STEP 5. Damaged region highlighted to the user
NEW STEP 6. The user decides that the details are enough
STEP 8. The user decides that the desired damaged regions in the ceilings are correctly detected
STEP 9. Output the results and
STEP 11. Save for future teaching dataset

Fig. 4.17 Showcase 1 for Fig. 4.16(a)

From the showcase in Fig. 4.17, it can be found that:

1. The digit prediction to the original image increases from 0.903 to the prediction to the cropped image as 0.963, which indicates that the precision increases.

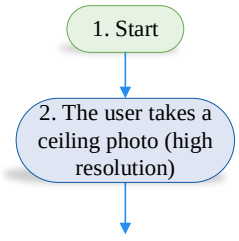
2. The saliency map and the grad-CAM may show some misleading highlighted regions (the weaker highlighted regions to the original image in this showcase STEP 5), but the user can decide to ignore the regions obviously unrelated to the ceilings.

3. The saliency map and the grad-CAM show more detailed highlight regions to the cropped image than those to the original image.

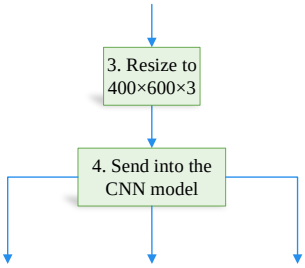
Through the interactivities shown above, the ceiling damage detection system aids the user to detect damages in the ceilings.

2. Showcase 2 (Fig. 4.16(b-1) and Fig. 4.16(b-2)):

In Showcase 2, the second ceiling image (Fig. 4.16(b-2)) is taken from a much nearer perspective than the first one (Fig. 4.16(b-1)). Firstly, it seems there are no damages from the first image by human, but the system reports that there are damages in the first image and revealed highlighted regions. By taking another photo to the highlighted regions (Fig. 4.16(b-2)), the damages in the ceilings are detected. Through this process, the ceiling damage detection system detects the damage before the user does and points out the damaged regions for the user.



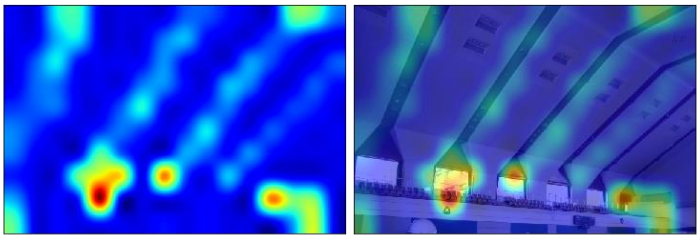
STEP 1&2. Start from the original ceiling image
(resolution: 2161×1316×3)



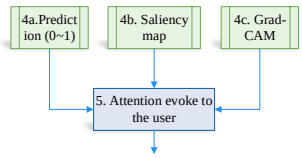
STEP 3. Resize to resolution: 600×400×3 and STEP 4. send to the CNN model

4a. Prediction: 0.787

4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image



STEP 5. Possible damaged region highlighted to the user

STEP 6. Although it seems there are no damages in Fig. 4.16(b-1), the user makes comprehensive considerations to the outputs and decides to see more details on lower left region of Fig. 4.16(b-1) and to take another photo

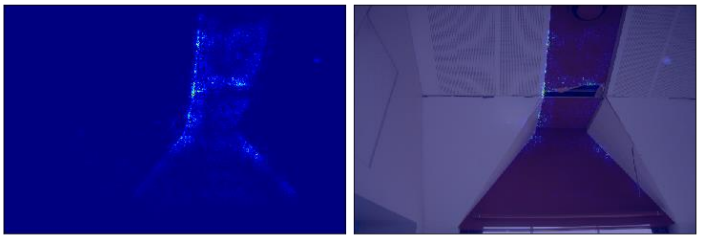
STEP 7b. The user takes another photo to the lower left region of Fig. 4.16(b-1) (resolution: 2592×1944×3)

NEW STEP 2. The user takes a new ceiling photo from a nearer standing point

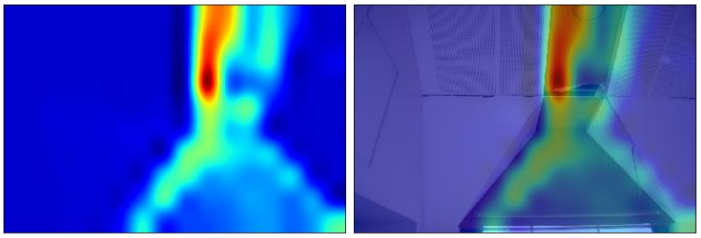
NEW STEP 3. Resize the cropped image to resolution: 600×400×3 and

NEW STEP 4. Send to the CNN model again

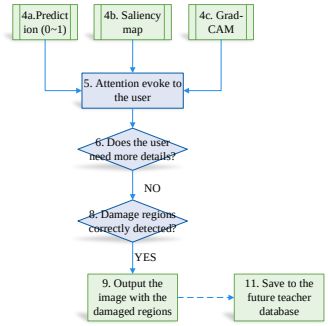
4a. Prediction: 0.910



4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image



NEW STEP 5. Damaged region highlighted to the user

NEW STEP 6. The user decides that the details are enough

STEP 8. The user decides that the desired damaged regions in the ceilings are correctly detected;

STEP 9. Output the results and

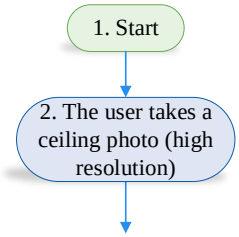
STEP 11. Save for future teaching dataset

Fig. 4.18 Showcase 2 for Fig. 4.16(b)

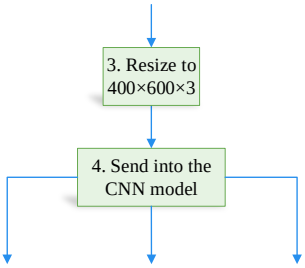
In Fig. 4.18, the ceiling image (Fig. 4.16(b-1)) seems intact at the first glance by the user. While the damaged region is correctly detected by the ceiling damage detection system through the interactive performance between the user and the system. This indicates that the system aids the user to detect damages when they are difficult to find from the first glance.

3. Showcase 3 (Fig. 4.16(c-1) and Fig. 4.16(c-2)):

For Fig. 4.16(c-1) and Fig. 4.16(c-2), in fact the system has made incorrect predictions for mistaking a speaker as a damaged region. The interactive process is shown in Fig. 4.19:

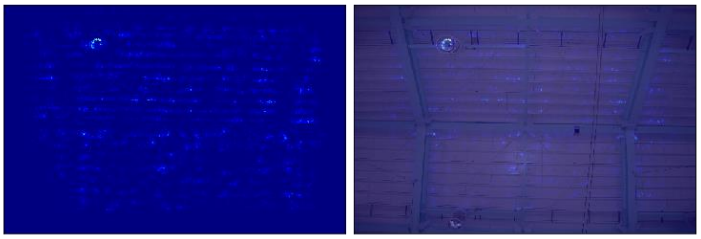


STEP 1&2. Start from the original ceiling image
(resolution: 2048×1536×3)

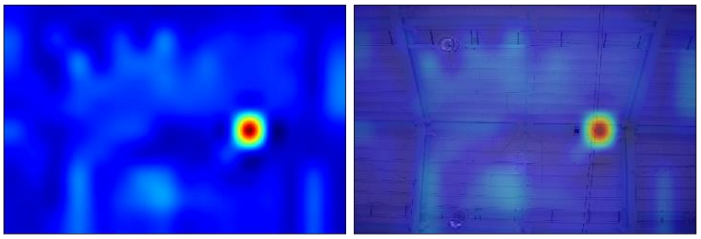


STEP 3. Resize to resolution: 600×400×3 and STEP 4. send to the CNN model

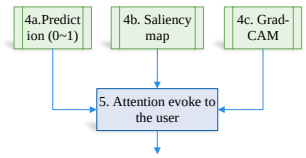
4a. Prediction: 0.580



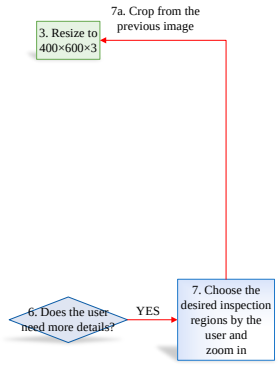
4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image

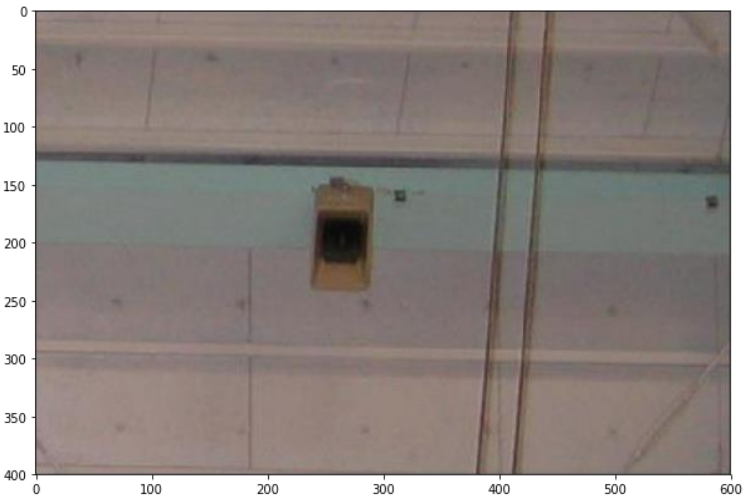


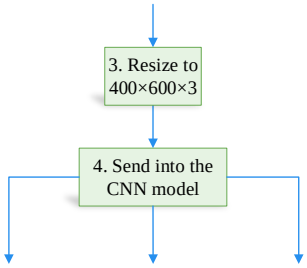
STEP 5. Damaged region highlighted to the user



STEP 6. The user makes comprehensive considerations to the outputs and decides to see more details on the strongly highlighted region in the grad-CAM and ignores the weaker highlighted regions

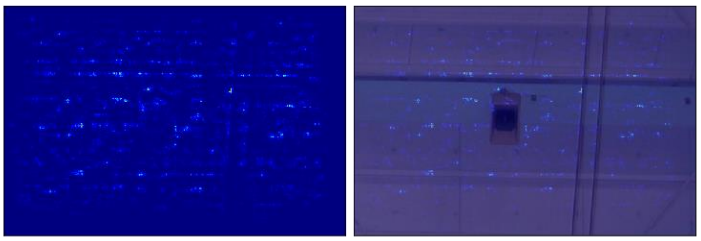
STEP 7a. Crop the highlighted regions by the user from the original image (resolution: 412×275×3)



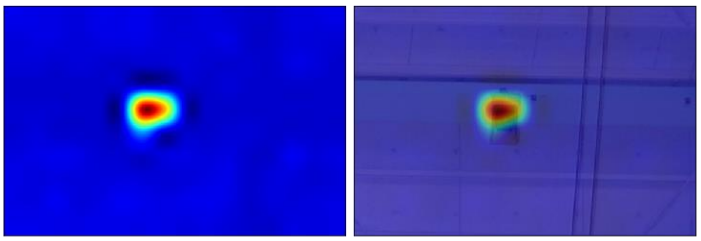


NEW STEP 3. Resize the cropped image to resolution: 600×400×3 and
NEW STEP 4. Send to the CNN model again

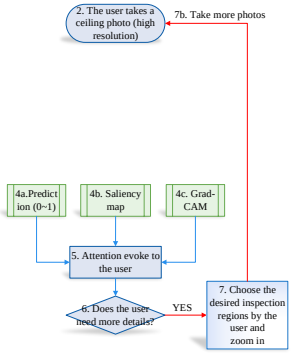
4a. Prediction: 0.620



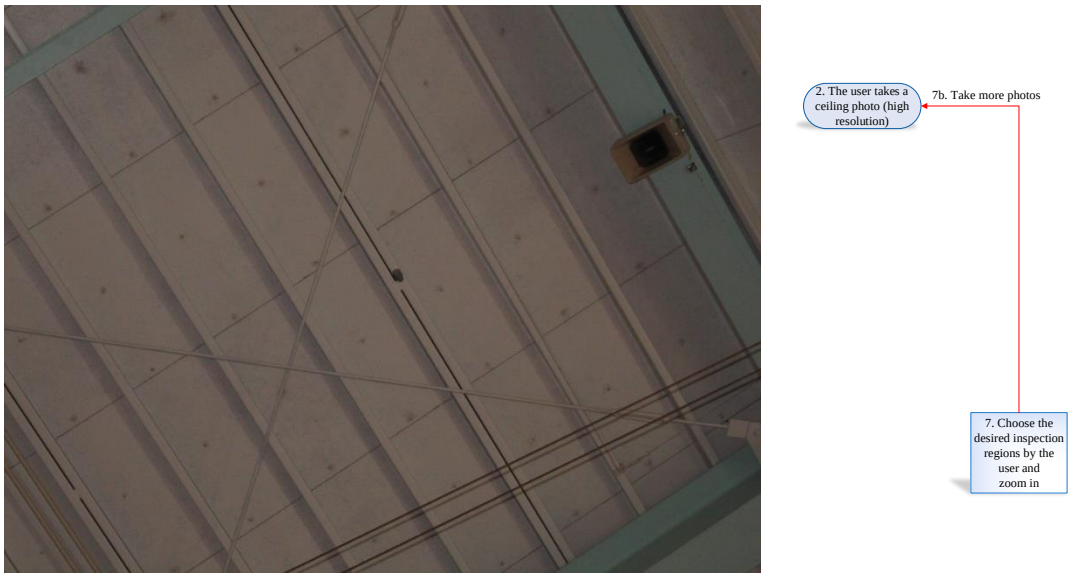
4b. The saliency map and overlap the saliency map to the ceiling image



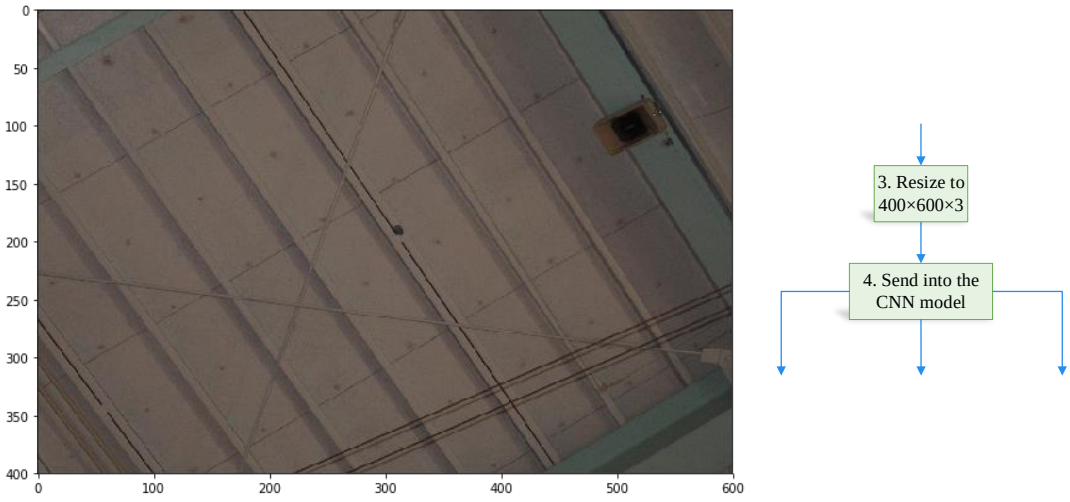
4c. The grad-CAM and overlap the grad-CAM to the ceiling image



NEW STEP 5. Damaged region highlighted to the user
NEW STEP 6. The user finds out that the highlighted region is not damaged ceiling but wants more information, the user decides to take another photo to the ceilings

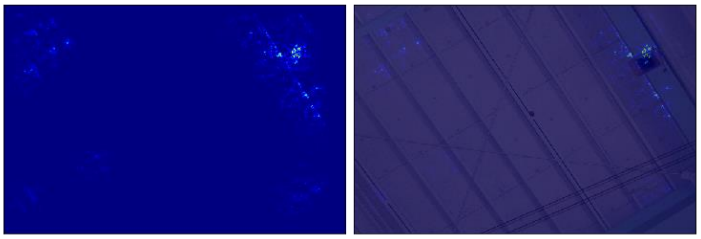


STEP 7b. Take a new photo to the ceilings
(resolution: 2048×1536×3)
NEW STEP 2. The user takes a new ceiling photo

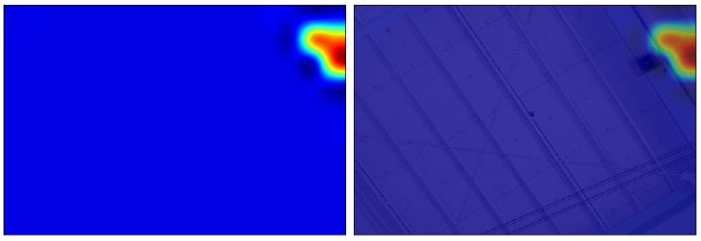


STEP 3. Resize to resolution: 600×400×3 and STEP 4. send to the CNN model

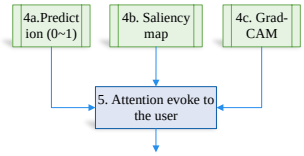
4a. Prediction: 0.684



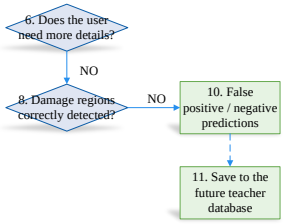
4b. The saliency map and overlap the saliency map to the ceiling image



4c. The grad-CAM and overlap the grad-CAM to the ceiling image



STEP 5. Possible damaged region highlighted to the user



STEP 6. The user finds that this highlighted region is a special formed speaker that is mistaken by the CNN model as damaged region.

The user decides that the details are enough.

STEP 8. The user decides that the ceilings are intact and the CNN model has made incorrect predictions

STEP 10. The image is labeled as intact (False positive)

STEP 11. Save for future teaching dataset

Fig. 4.19 Showcase 3 for Fig. 4.16(b-1) and Fig. 4.16(b-2)

In Fig. 4.19, the Showcase 3 clarifies the procedure when the system makes incorrect prediction. Although incorrect predictions may reduce the user confidence for the ceiling damage detection system, incorrect predictions occur for reasons. In Showcase 3, the special formed speaker is mistaken by the system as a damaged region for the CNN model has not been taught by such form of object. In fact, it is difficult for human to make correct judgement at the first glance of it: it really looks like a hole in the ceilings.

At the end of Showcase 3, the ceiling image is saved for future training dataset. Collecting new ceiling images, especially those that the CNN model makes incorrect predictions to are crucial for the further development of the CNN model.

4.4 Conclusion

In this chapter, the CNN model built and trained in Chapter 3 is investigated and interpreted by visualization methods (the intermediate convolutional layer outputs, the activation maps to the filters, the saliency map and the Grad-CAM) and a ceiling damage detection system with user-CNN interactive process is proposed. The conclusions are:

1. Visualizations to the intermediate convolutional layer outputs for given ceiling images prove that the filters in the convolutional layers do notice the ceiling and non-ceiling regions in the images (shown in Fig. 4.1).

2. The generations of the images that most activating the filters in the convolutional layers prove that the CNN model performs gradual abstractions through the convolutional layers (shown in Fig. 4.3, the images become more and more complex).

3. The saliency map method and the Grad-CAM method confirm that the CNN models have learnt to recognize the features representing the ‘intact’ and ‘damaged’ ceilings firstly. Furthermore, they successfully support ceiling damage detection function from the inner process perspective interpretation to the CNN model (shown in Fig. 4.5 and Fig. 4.12).

4. A ceiling damage detection system with user-CNN interactive process (Interactive-AI) is proposed and has been proved feasible because: **a.** The user can either be a well-

trained ceiling expert who performs routine inspection or a layman (refugee) who needs to know the working conditions of the ceilings hanging over his / her head during a disaster circumstance; **b.** the user understands what is happening and knows what the ceiling damage detection system is doing in the damage detection process; **c.** the accuracy is increased and more damages are detected through the interactive process; **d.** this interactive system can minimize the bad effects of the incorrect predictions; **e.** this system is designed to collect more ceiling images for further improvements (shown in Fig. 4.20 and Fig. 4.21).

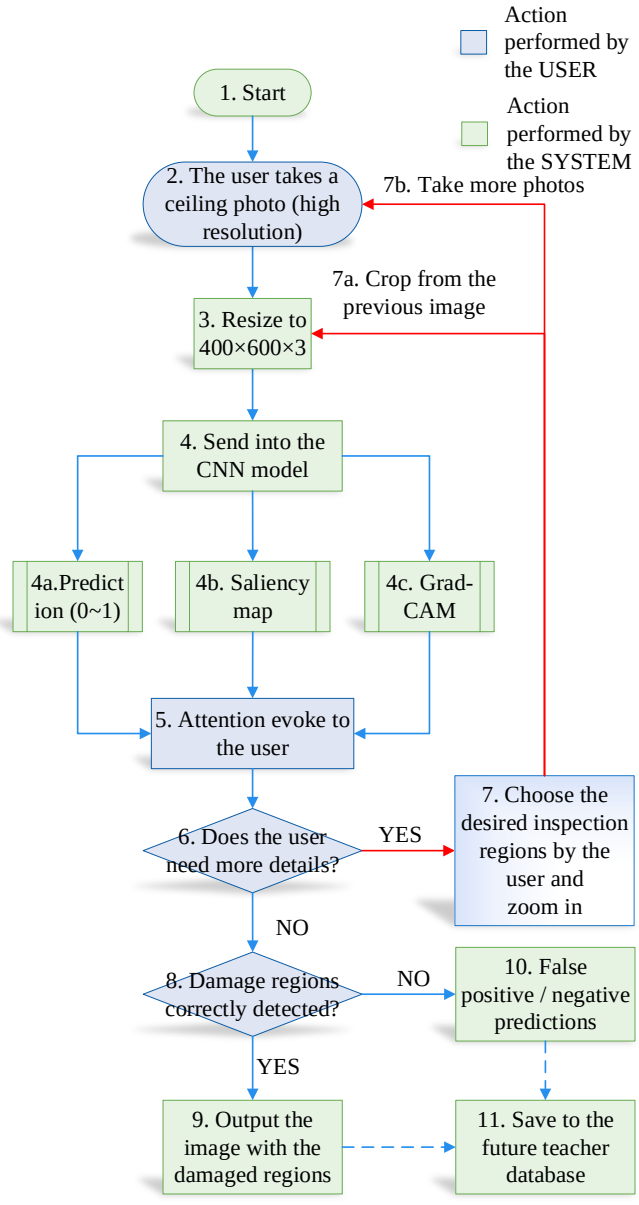


Fig. 4.20 The ceiling damage detection system

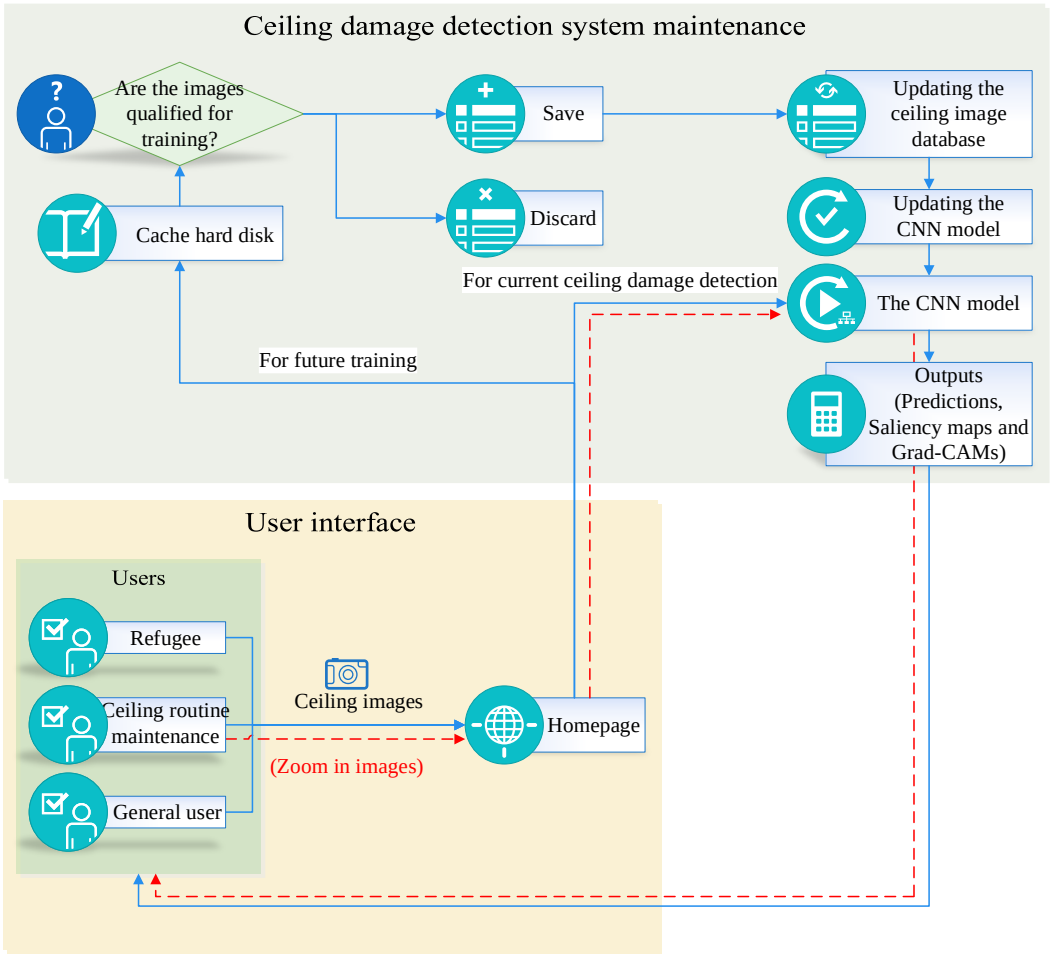


Fig. 4.21 A web-based ceiling damage detection system

5. Transfer Learning for Ceiling Damage Detection

5.1 Introduction to transfer learning

5.1.1 A brief theory of transfer learning

Since deep learning is a branch of artificial intelligence, a high-performance deep learning model revealing intelligence behaviors has very strong learning potential to grasp a related domain that is close to what it has learnt. This is the motivation for transfer learning. Transfer learning dates back to decades ago, with the intention to build lifelong machine learning methods that keep and reuse previously learnt knowledge [142]. Especially when the training data in the desired domain is very difficult or expensive to obtain, there comes out the need to build a high-performance learning model that can learn as much as possible even if the training data is rare [143]. This is also the reason for introducing transfer learning into ceiling damage detection task.

With the big data age emerging, in many domains there are database storages relating to but not the same as the database that a well-performance deep learning model was trained on. However, these databases storages are usually not big or diverse enough to support the training requirements to a deep learning model to be built and taught from scratch. Moreover, these databases could be rare, expensive to collect or label. The idea of transfer learning has provided successful solutions to many such domains such as sentiment analysis [144], image classification [145], medical diagnosis [110].

The definition of transfer learning can be described as follows [143, 146-149]:

A domain D is defined by a feature space X and a marginal probability distribution $P(X)$. A task T is defined as: for a given domain D, a label space Y and a predictive function $F(\cdot)$ that fits best with $F(X)=Y$. Then the transfer learning is: for a given source domain D_s with a corresponding source task T_s and a target domain D_t with a corresponding task T_t, the process of training the target predictive function $F_t(\cdot)$ by using D_s and T_s. The source domain D_s can be extended to multiple source domains.

5.1.2 Transfer learning for image classification

In the domain of image classification, the transfer learning is to use the state-of-the-art image classification models trained on the ImageNet database [71] to adept to a new

image database instead of from being built from scratch. Fig. 5.1 shows the basic transfer learning idea for image classification. The convolutional layers of the pre-trained models are frozen to perform abstractions in the inputs; the following fully connected layer are trained based on the inputs. The specific methods adopt in transfer learning depend on two factors: 1. the size of the new image database; 2. the similarity of the new image database to the ImageNet database. In different combinations of the factors, there are four cases in transfer learning for image classification:

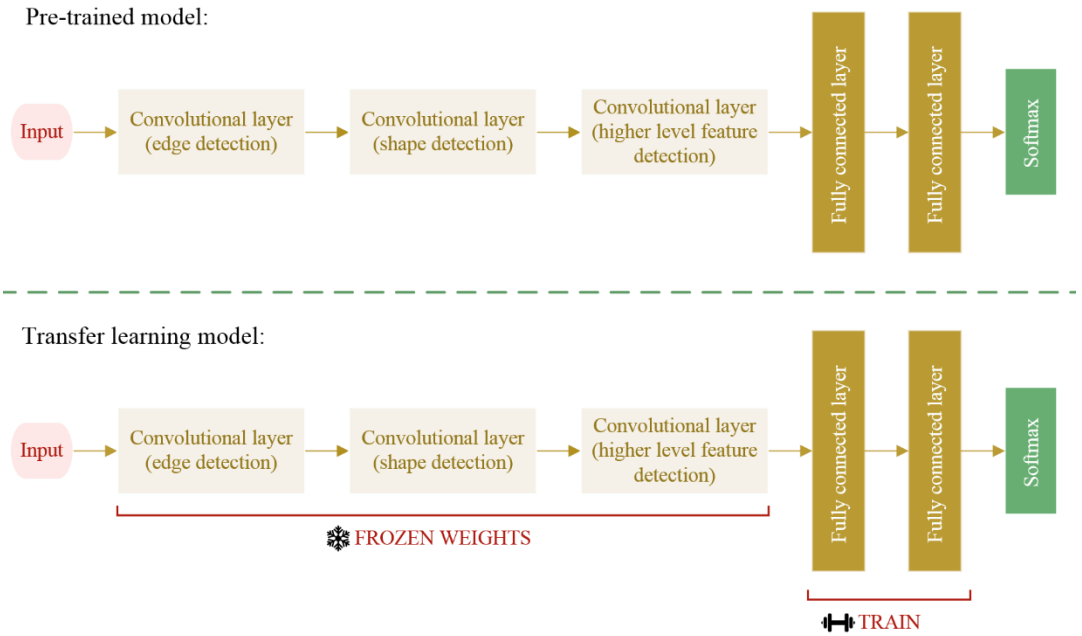


Fig. 5.1 Basic transfer learning for image classification

1. **SMALL** new database, **LOW** similarity of the new image database to the ImageNet database;

2. **BIG** new database, **LOW** similarity of the new image database to the ImageNet database;

3. **SMALL** new database, **HIGH** similarity of the new image database to the ImageNet database;

4. **BIG** new database, **HIGH** similarity of the new image database to the ImageNet database.

The boundaries between a 'big' database and a 'small' one could be subjective. A dataset with one million images is definitely big and a dataset with one thousand images is really small. The boundaries between similarities are subjective as well. The front view and the side view of the same object could be totally different. The key is to

prevent overfitting in the transfer learning [145, 150].

For the four cases above, different methods to generate best-performance models using transfer learning are different:

1. SMALL new database, LOW similarity:

- Keep most of the layers directly connecting to the beginning of the pre-trained CNN networks, remove the rest of the pre-trained networks from the latter convolutional layers;

- Add a new fully connected layer to the remaining networks. The output of the new fully connected layer matches the class number of the new database;

- Randomize the weights in the new fully connect layer and freeze all the weights in the pre-trained networks;

- Train the new networks.

The slicing off the layers from the latter convolutional layer is because the new database is very different from the pre-trained networks that they do not share the same weights in the high level convolutional layers. The transfer learning in such circumstance only keeps the lower level features. The reason to freeze the weights in the pre-trained networks is to prevent overfitting due to the small new dataset.

2. BIG new database, LOW similarity:

- Remove the last fully connected layer and add a new fully connected layer matching the class number of the new database with randomly initialized weights;

- Train the whole networks from randomly initialized weights;/ or: Train the whole networks with the weights initialized from the pre-trained networks.

In this case, we have enough data to train the weights in the new networks, so there is no need to freeze any weights. Initializing the weights from the pre-trained networks may cost less time to train the model.

3. SMALL new database, HIGH similarity:

- Remove the last fully connected layer and add a new fully connected layer

matching the class number of the new database with randomly initialized weights;

- Randomize the weights in the new fully connect layer and freeze all the weights in the pre-trained networks;

- Train the new networks.

The images in the new database are similar with those images in the pre-trained networks. They share most of the weights in the convolutional layers except for the number of class. To avoid overfitting, the weights in the pre-trained networks are frozen in the training process.

4. BIG new database, HIGH similarity:

- Remove the last fully connected layer and add a new fully connected layer matching the class number of the new database with randomly initialized weights;

- Randomize the weights in the new fully connect layer and initialize the rest of all the weights from the pre-trained networks;

- Train the new networks.

This is the most favorable one in these four circumstances. There is not much to worry about overfitting because of big amount of data. The new dataset and the pre-trained networks dataset share the same high-level features that the weights could be initialized from the pre-trained networks.

Most of the prevailing pre-trained models are trained based on the ImageNet database which is a huge collection of daily objects with 1000 classes. Although the ImageNet database collects so many objects, the research object in this thesis is the damage detection in ceilings, which is very different from the ImageNet database. The transfer learning in this thesis best conforms with the Case 1: small new dataset and low similarity, which is also the most unfavorable condition.

5.1.3 Pre-trained models: VGG16 and VGG19 [134]

In 2014, Simonyan and Zisserman who were affiliated with Visual Geometry Group (VGG) developed the VGGNet series using only convolutional layers, max pooling

layers and fully connected layers. The main feature in the VGGNet series is that the convolutional layers use only 3×3 filters to perform down-sampling to the inputs. Table 5.1 shows the details of the configuration to the VGGNet series. There are five models in the VGGNet series (from A to E in Table 5.1). VGG16 and VGG19 refer to the model D and E with corresponding number of layers. With the number of convolutional layers growing, the trainable parameters also increase (shown in Table 5.2). The architecture of VGG16 and VGG19 are shown in Fig. 5.2.

Table 5.1 VGGNet series configuration [134]
(conv<filter size>-<number of filters>)

A	A-LRN	B	C	D (VGG16)	E (VGG19)
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input: 224×224×3					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
max pooling					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
max pooling					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
max pooling					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
max pooling					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
max pooling					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 5.2 Number of trainable parameters (in millions) [134]

Network	A, A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

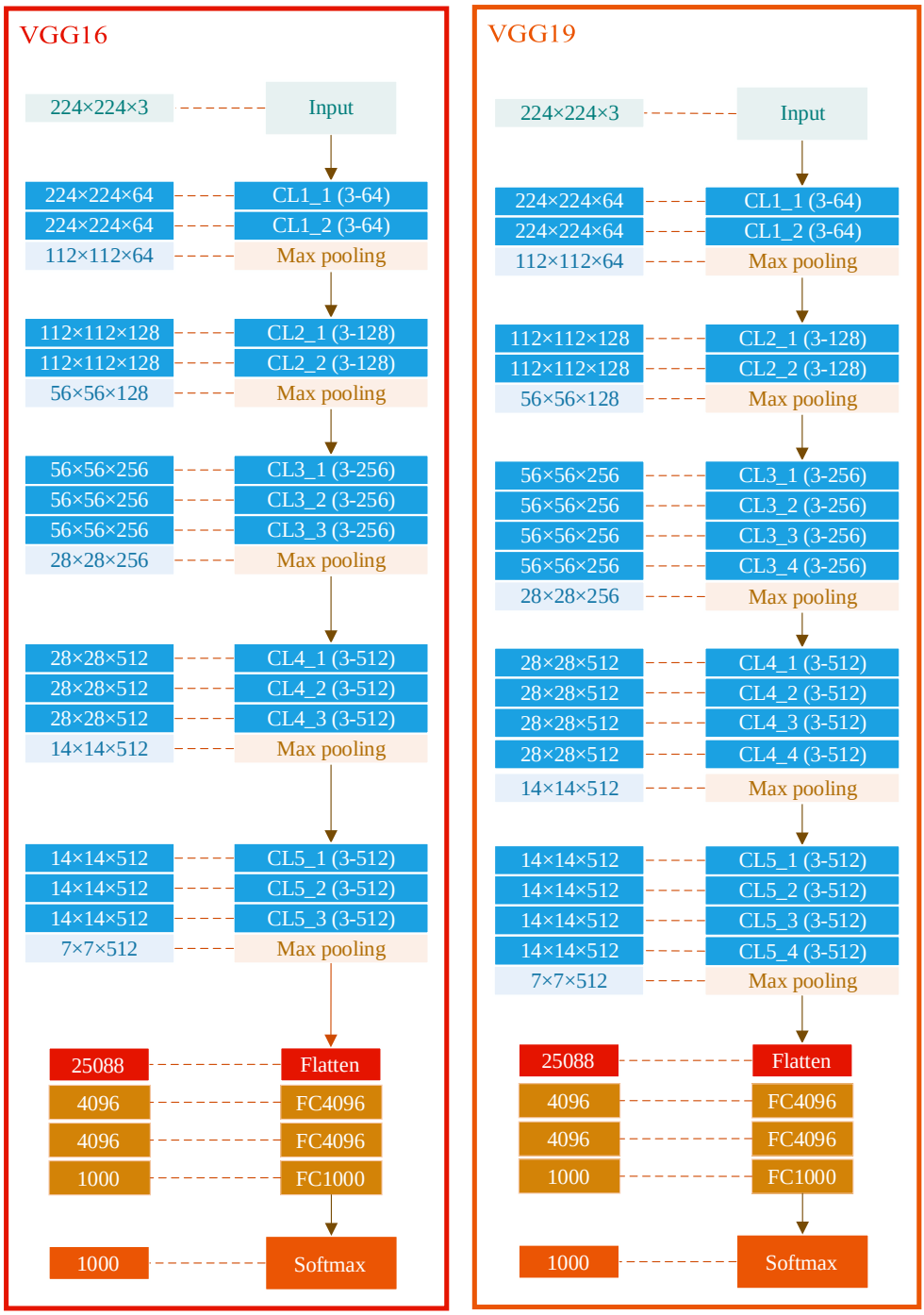


Fig. 5.2 The architecture of VGG16 and VGG19

The input to the VGGNet series is a fixed-size 224×224×3 image. The filters are set as a unified receptive field of 3×3 (in model C, at each end of a convolutional block the filter is set as 1×1). The stride of convolutional layers is 1, the stride of the max pooling layers is 2 with a 2×2 kernel. The networks were trained on a system with four NVIDIA Titan Black GPUs for 2-3 weeks respectively depending on the architecture of the models. The performance of the classification task is evaluated by the top-1 and top-5 error. The VGG team trained and validated the VGGNet models in multi scales of

images. Table 5.3 shows the performance of the VGGNet models.

Table 5.3 VGGNet performance at multiple test scales [134]

Model (Table 5.1)	Smallest image side		Top-1 error (%)	Top-5 error (%)
	Train (S)	Test (Q)		
B	256	224, 256, 288	28.2	9.6
C	256	224, 256, 288	27.7	9.2
	384	352, 384, 416	27.8	9.2
	[256; 512]	256, 384, 512	26.3	8.2
D	256	224, 256, 288	26.6	8.6
	384	352, 384, 416	26.5	8.6
	[256; 512]	256, 384, 512	**24.8**	**7.5**
E	256	224, 256, 288	26.9	8.7
	384	352, 384, 416	26.7	8.6
	[256; 512]	256, 384, 512	**24.8**	**7.5**

Nowadays, VGG16 and VGG19 are widely used in many other applications in image classification and recognition for the simplicity and easy-understandability. They are used as a baseline for feature extraction. The defect of the VGGNet is that the weights are very large to handle (the weight files for VGG16 is over 533MB and for VGG19 is over 574MB).

5.2 Building and training transfer learning models for ceilings damage evaluation

The first problem we are facing with is that the size of the training images of VGG16 and VGG19 models is 224×224×3 while the size of the training ceiling images in this research is 400×600×3. The first possible solution is to resize the ceiling images from 400×600×3 to 224×224×3 to generate a new ceiling dataset to fit the VGG models. However, such dramatic resize operation will lose many details in the original images. Another solution is originated from an important characteristic of convolutional neural networks: the filters scanning over an input matrix do not require a specific size. In other words, the same filters are possible to generate different sizes of outputs given different sizes of inputs.

The second problem is if the pre-trained filters are possible to perform proper abstractions to the ceiling images that were never shown before. Since the transfer learning of ceiling images recognition is very different from the ImageNet dataset, if the VGG16 and VGG19 models are possible to learn the features representing ‘intact’

and ‘damaged’ are unknown.

The third problem is to find a proper architecture for the tail part of the transfer learning models. The tails of VGG16 and VGG19 are fully connected layers that linearize the matrices generated from the 224×224×3 training data which is different from the 400×600×3 ceiling images. The tails of these models need to be modified for the specific task of ceiling image processing.

5.2.1 Building and training a transfer learning model using the VGG16 weights

According to the architecture of VGG16 and the transfer learning method, a new CNN model using the VGG16 weights for ceiling damage evaluation was built. We kept the convolutional blocks from the VGG16 model and added new layers like a GAP layer and fully connected layers to the remaining convolutional blocks. This model is named as a ‘TF_VGG16’ model with the architecture shown in Fig. 5.3. We froze the convolutional blocks till the CL5_2 (the weights in them are inherited from the original VGG16 weights) but left the last convolutional layer CL5_3 trainable. The reason for this is that the last convolutional layer processes the high-level feature abstraction to the training ceiling dataset. The training ceiling dataset was very different from the dataset VGG16 was trained. Leaving the last convolutional layer CL5_3 trainable helps the TF_VGG16 model dealing with the differences among the datasets. Fully connected layers and a dropout layer were connected to the trainable CL5_3 layer to perform final abstractions.

The training process was similar to that in Chapter 3: the input images are first processed through data augmentation. In the training process, 50 epochs were performed. The accuracy and loss curves to the epochs are shown in Fig. 5.4. Although the training accuracy increases with the epochs, the validation accuracy remains around at 0.9 since the 10th epoch, which indicates that the TF_VGG16 model begins to be overfit after the 10th epoch. The loss-epoch curve also indicates that the TF_VGG16 model begins to be overfit after the 10th epoch since the validation loss begins to increase with the epoch increases. The weights at the 10th epoch were saved as the final weights of the TF_VGG16. The final prediction accuracy to the test dataset is 90.3%, which is a significant increase to the CNN model prediction of 86.2%. The sensitivity and the specificity are shown in Table 5.4.

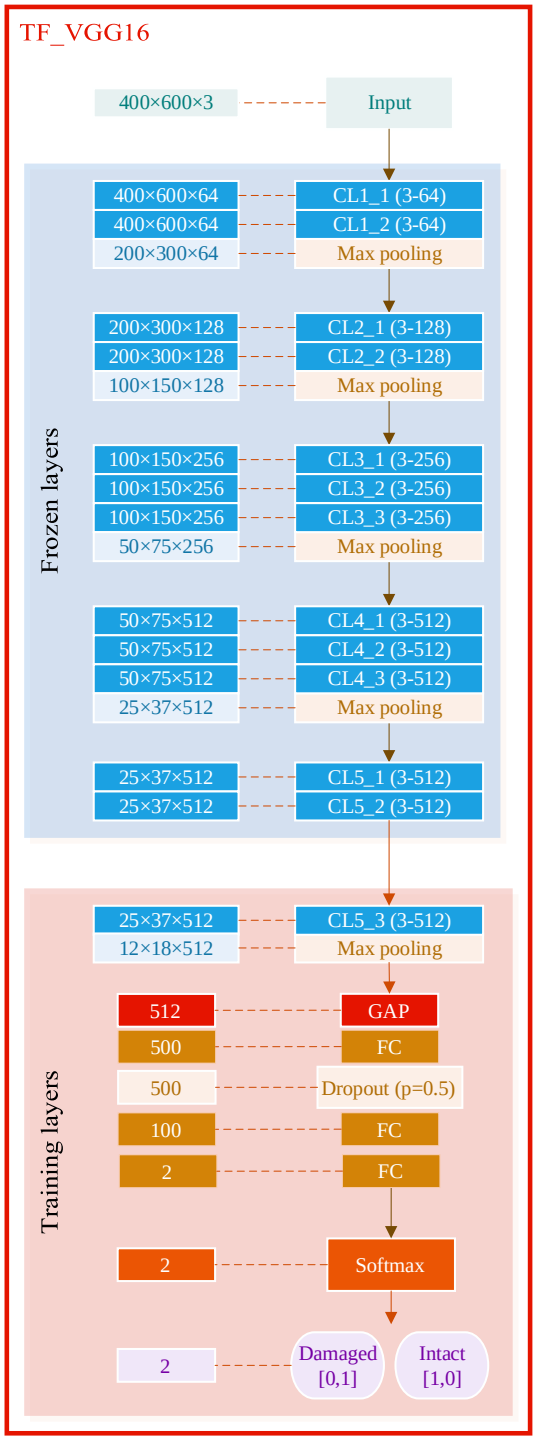


Fig. 5.3 The architecture of TF_VGG16

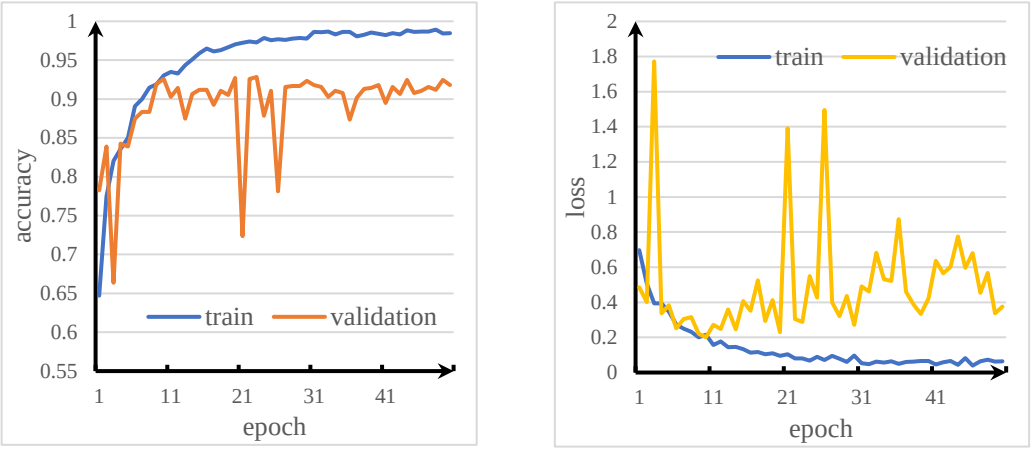


Fig. 5.4 The training accuracy and loss curves of TF_VGG16

Table 5.4 Sensitivity and specificity of TF_VGG16			
	Negative (0)	Positive (1)	Total
True	TN: 221	TP: 133	354
False	FN: 9	FP: 29	38
Total	230	162	392
Accuracy = (TP+TN) / SUM = 0.903 Sensitivity = TP / (TP+FN) = 0.937 Specificity = TN / (TN+FP) = 0.884			

5.2.2 Building and training a transfer learning model using the VGG19 weights

Although the architecture of VGG19 bears a close resemblance to that of VGG16, using the same transfer learning principles of TF_VGG16 to build another CNN model would result in poor convergence performance in the prediction accuracy. To the transfer learning model based on the stem of VGG19, the trainable layers are only in the tail layers without the convolutional layer. The transfer learning model is named as TF_VGG19, with the architecture shown in Fig. 5.5. The trainable parameters are the weights and biases in the full connected layers which can be represented by matrix multiplications.

The training process of TF_VGG19 was alike to that in TF_VGG16 with 50 epochs in total. The accuracy and loss curves to the epochs are shown in Fig. 5.6. The lowest validation loss occurred at the 45th epoch and the weights at this epoch was saved as the TF_VGG19 model. The final prediction accuracy to the testing dataset is 88.3%,

which is a slightly increase to the CNN model prediction of 86.2%. Table 5.5 shows the sensitivity and the specificity.

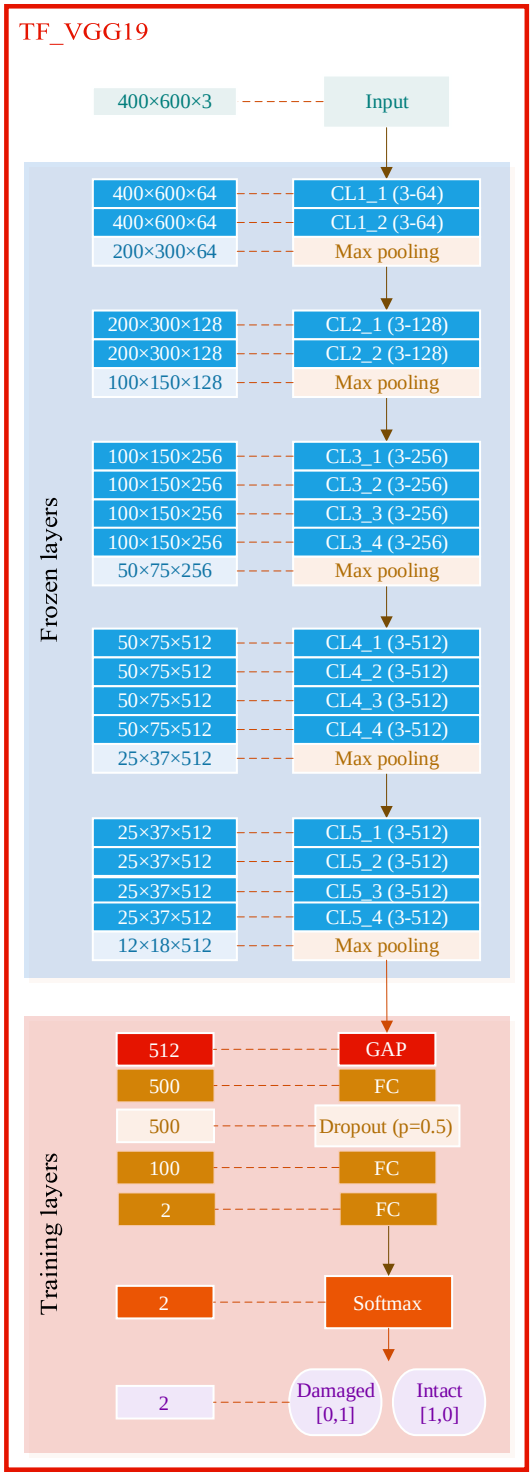


Fig. 5.5 The architecture of TF_VGG19

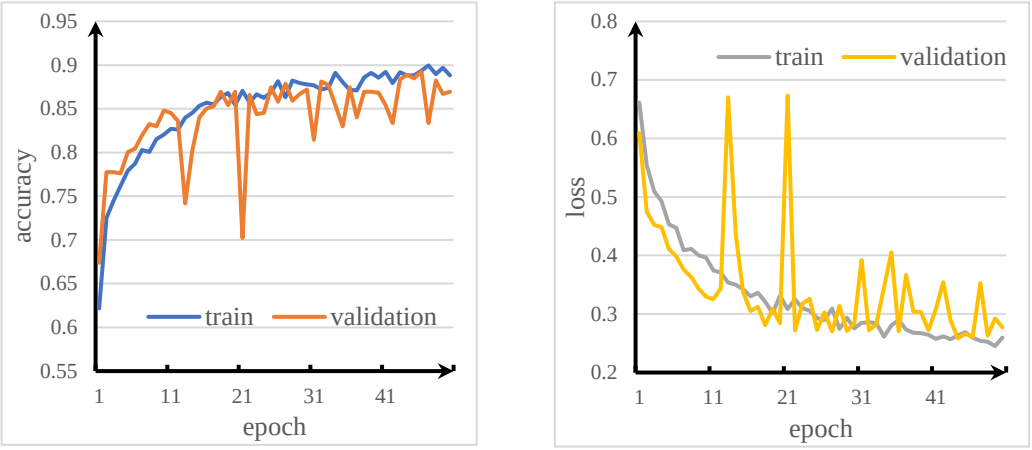


Fig. 5.6 The training accuracy and loss curves of TF_VGG19

Table 5.5 Sensitivity and specificity of TF_VGG19			
	Negative (0)	Positive (1)	Total
True	TN: 217	TP: 129	346
False	FN: 13	FP: 33	46
Total	230	162	392
Accuracy = (TP+TN) / SUM = 0.883 Sensitivity = TP / (TP+FN) = 0.908 Specificity = TN / (TN+FP) = 0.868			

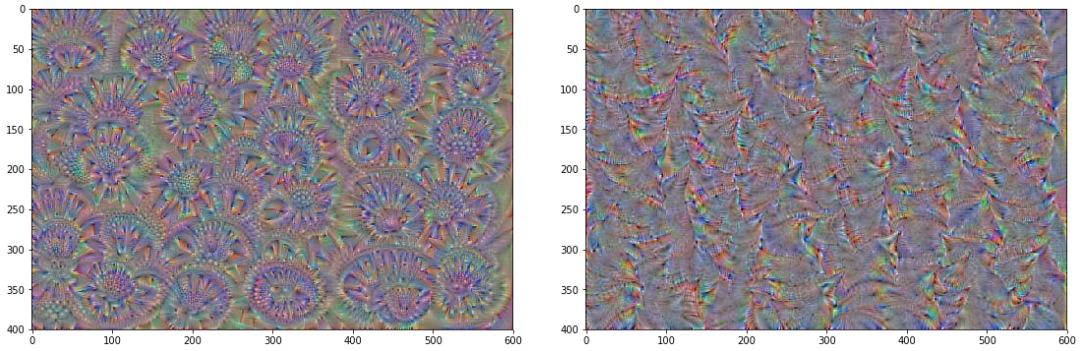
5.3 Evaluating and visualizing the transfer learning models

Both TF_VGG16 and TF_VGG19 achieved higher accuracy to the testing dataset that the CNN model built from scratch. If the transfer learning models did learn the representative characters of intact and damaged ceilings, the digitalized prediction, the saliency map and the Grad-CAM to the ceiling images that are never shown to these two models would match each other. In this section, the evaluating and visualizing methods like the final predictions, the notions of intact and damaged ceilings learnt by the transfer learning models (the images that most activate the final two nodes of prediction), the saliency maps and the Grad-CAM to the testing dataset and ceiling images from internet, are shown to investigate the transfer learning models.

5.3.1 The TF_VGG16 model

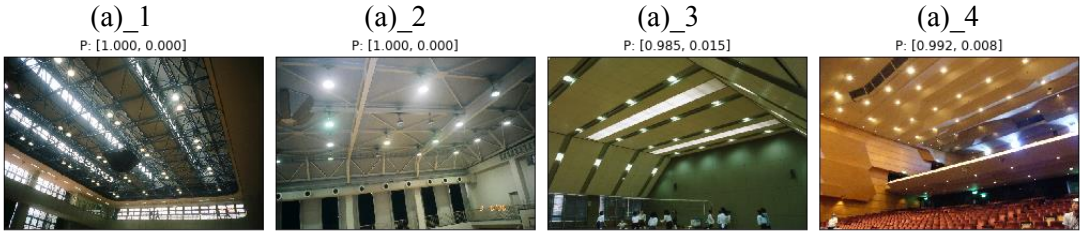
The final prediction accuracy to testing dataset is 90.8%. The images most activating the nodes of 'intact' and 'damaged' classes in TF_VGG16 are shown in Fig. 5.7. The images reflecting the notions of 'intact' and 'damaged' may seem a little abstract at the first glance, but they do extract the most important features of the two classes: the 'intact' ceilings usually contain shining lights orderly arranged like the divergent circles in Fig. 5.7(a); the 'damaged' ceilings usually contain irregular jagged tearing marks shown in Fig. 5.7(b).

The final predictions, the saliency maps and the Grad-CAM to the images selected from the testing dataset (a) and the internet (b) are shown in Fig. 5.8. The accuracy to the ceiling images from internet is 91.1%. The saliency maps in TF_VGG16 are much clearer and reflect more information about the ceilings.

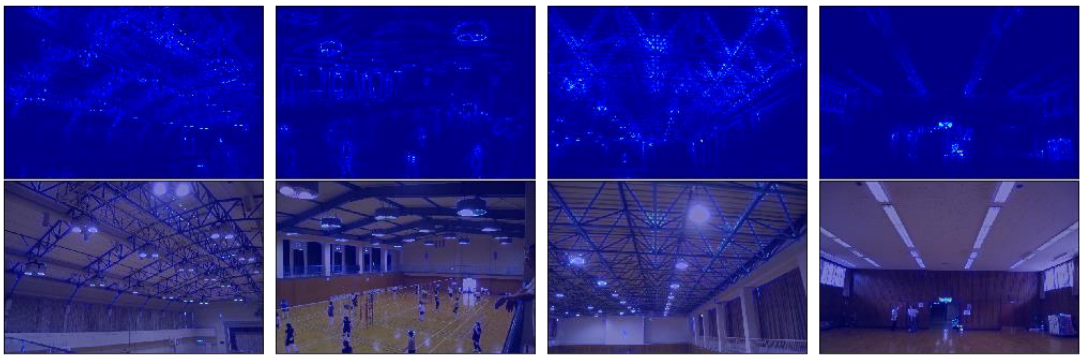


(a) intact (b) damaged

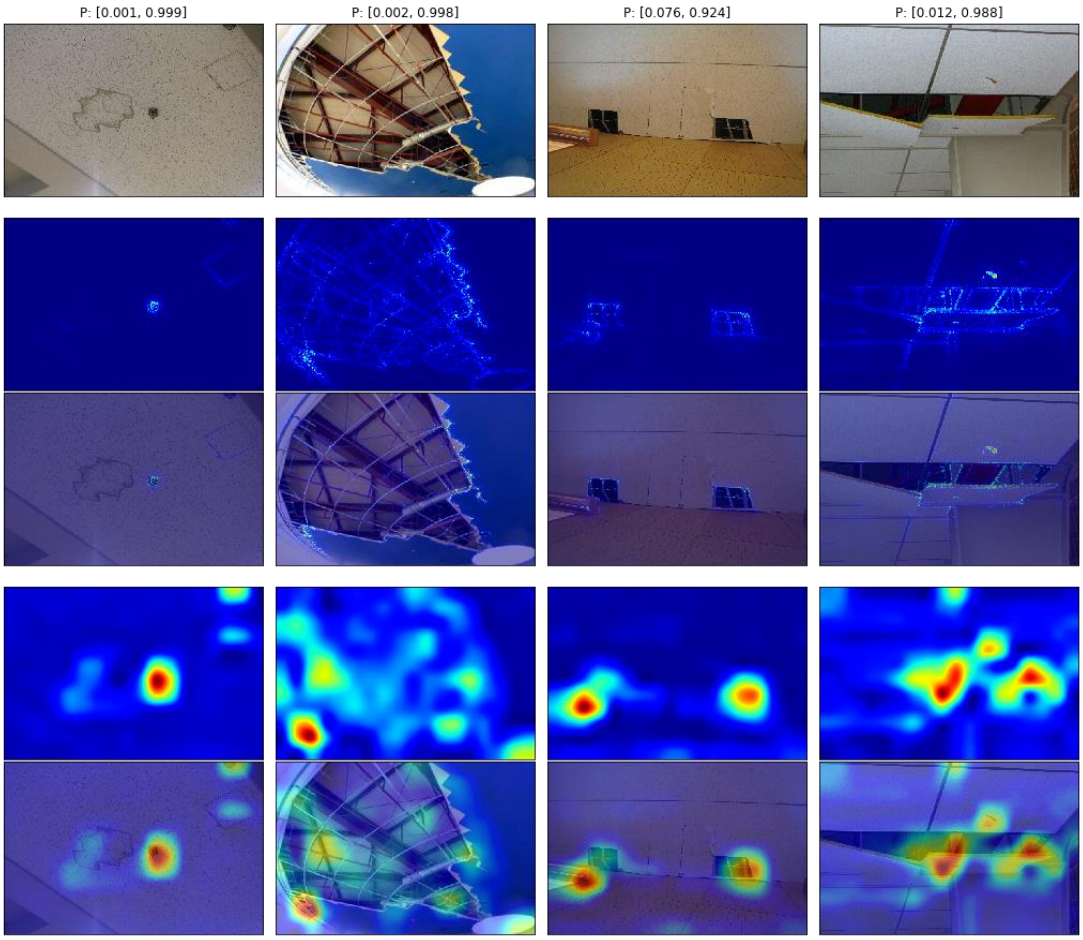
Fig. 5.7 'Intact' and 'damaged' images most activating TF_VGG16



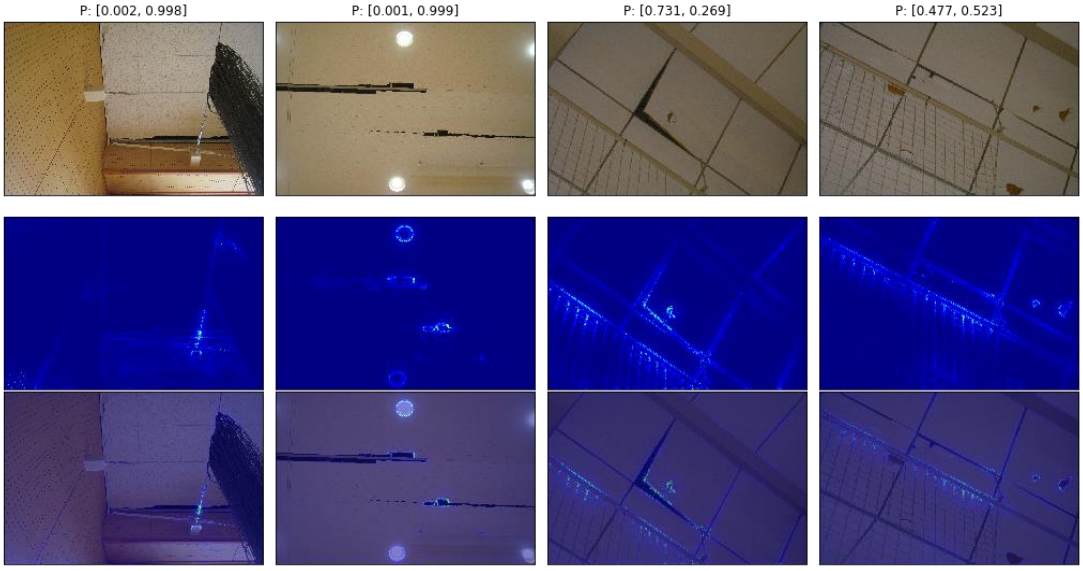
(a)_5 (a)_6 (a)_7 (a)_8

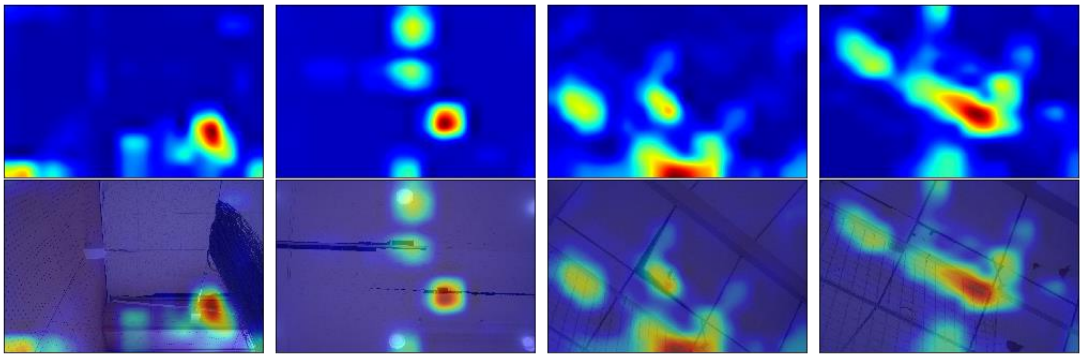


(a)_9 (a)_10 (a)_11 (a)_12



(a)_13 (a)_14 (a)_15 (a)_16





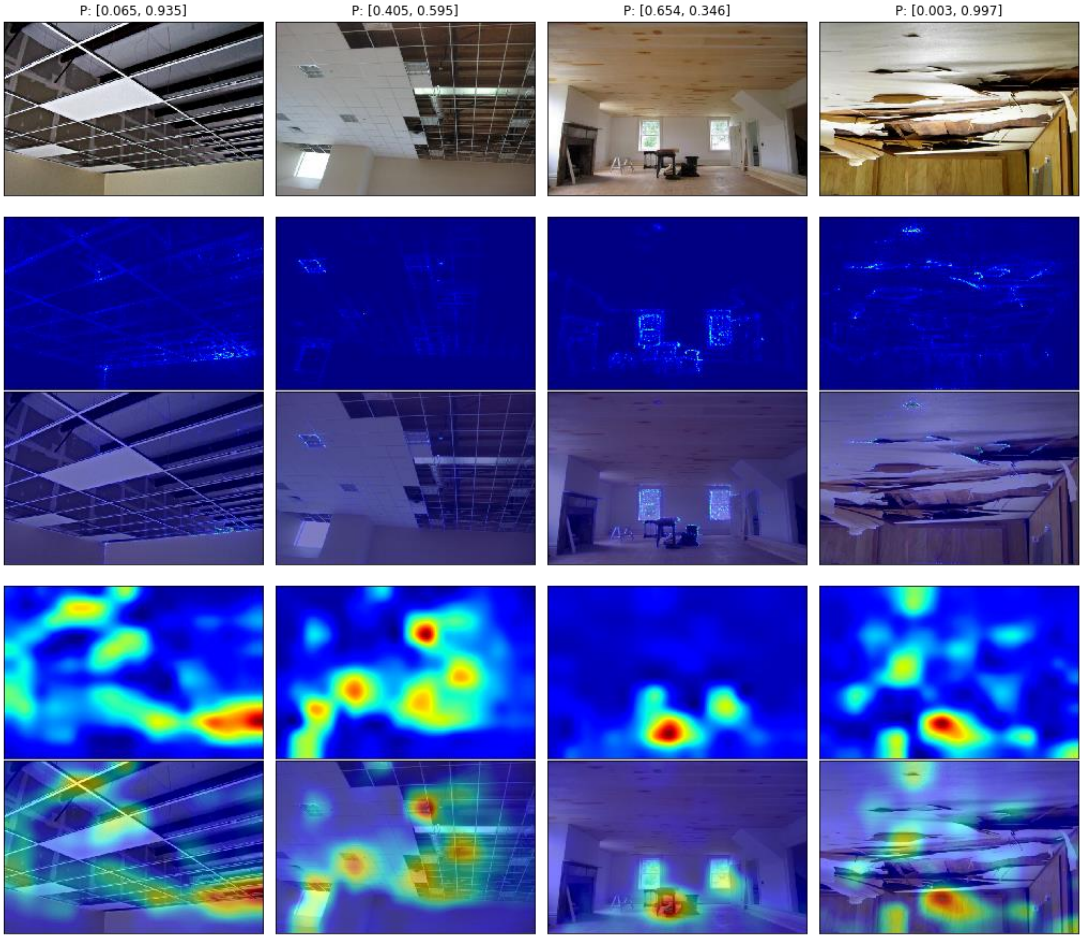
(a) testing dataset

(b)_1 P: [0.065, 0.935]

(b)_2 P: [0.405, 0.595]

(b)_3 P: [0.654, 0.346]

(b)_4 P: [0.003, 0.997]



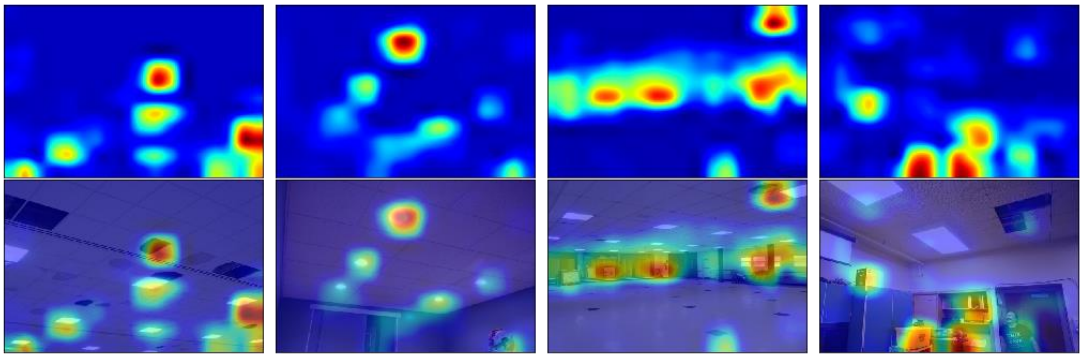
(b)_5 P: [0.843, 0.157]

(b)_6 P: [0.944, 0.056]

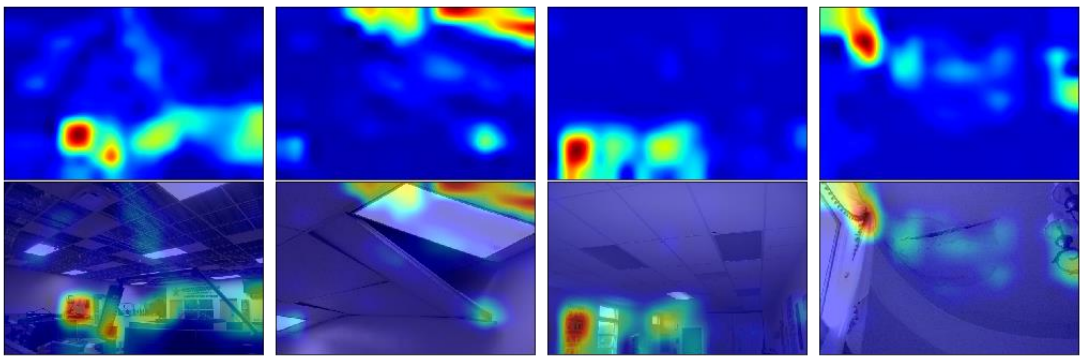
(b)_7 P: [0.813, 0.187]

(b)_8 P: [1.000, 0.000]

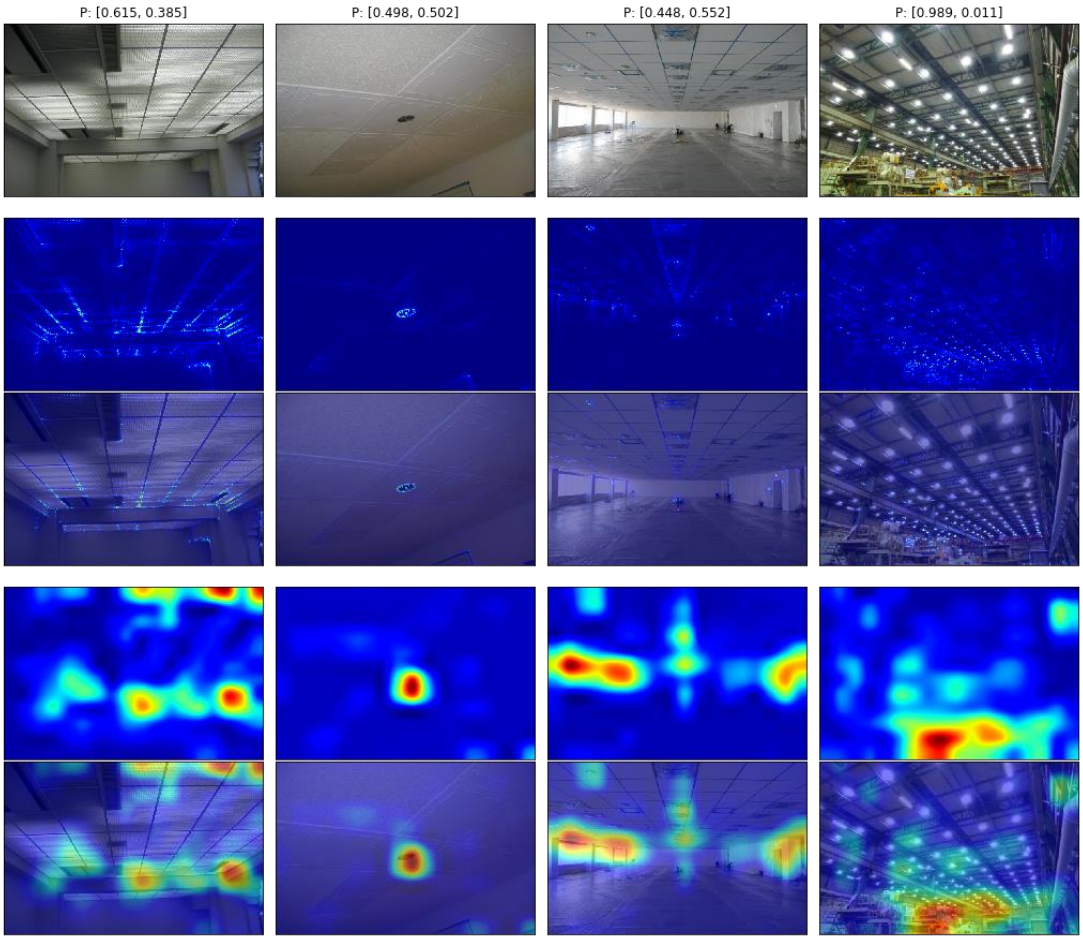
(b)_9 (b)_10 (b)_11 (b)_12



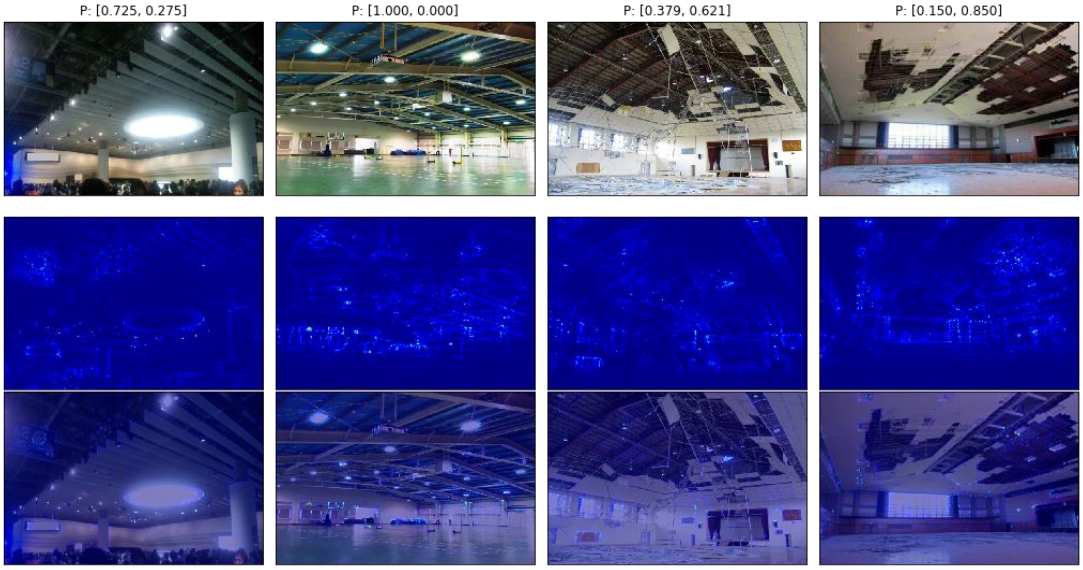
P: [0.999, 0.001] P: [0.180, 0.820] P: [0.871, 0.129] P: [0.000, 1.000]

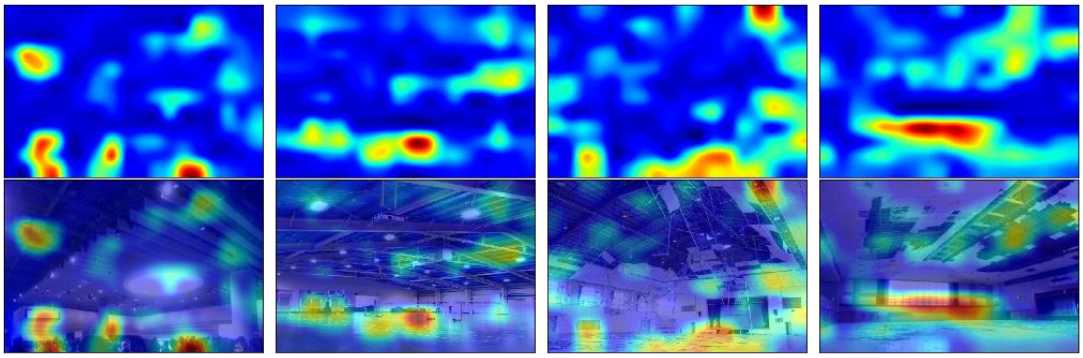


(b)_13 (b)_14 (b)_15 (b)_16



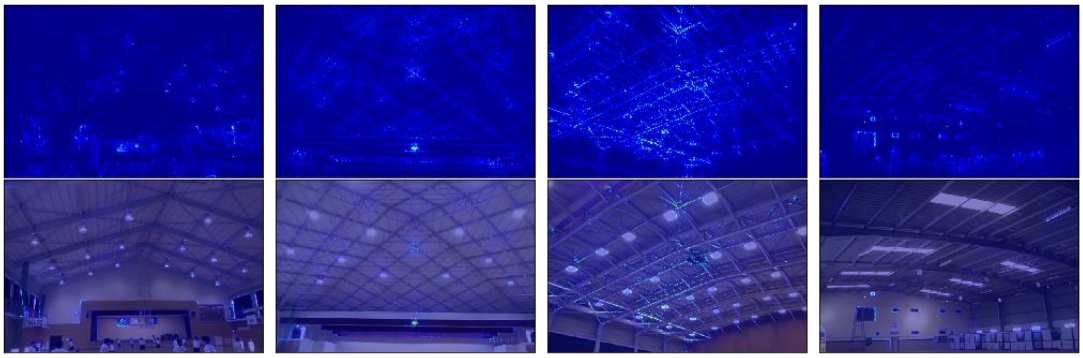
(b)_17 (b)_18 (b)_19 (b)_20

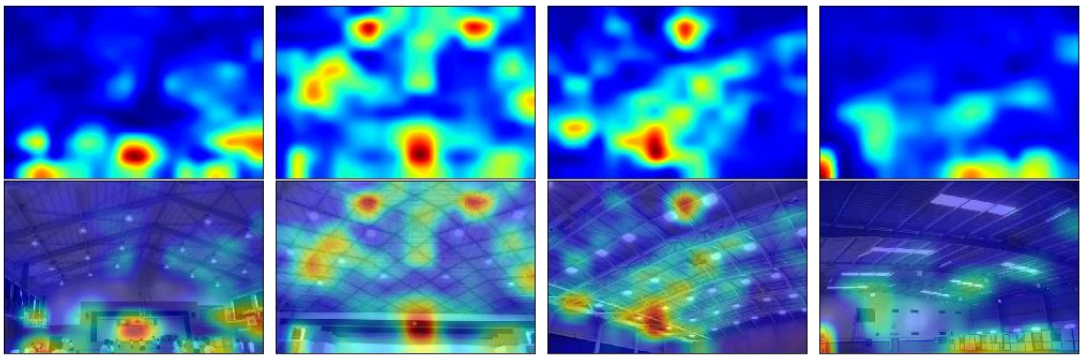




(b)_21 (b)_22 (b)_23 (b)_24

P: [1.000, 0.000] P: [1.000, 0.000] P: [0.999, 0.001] P: [1.000, 0.000]

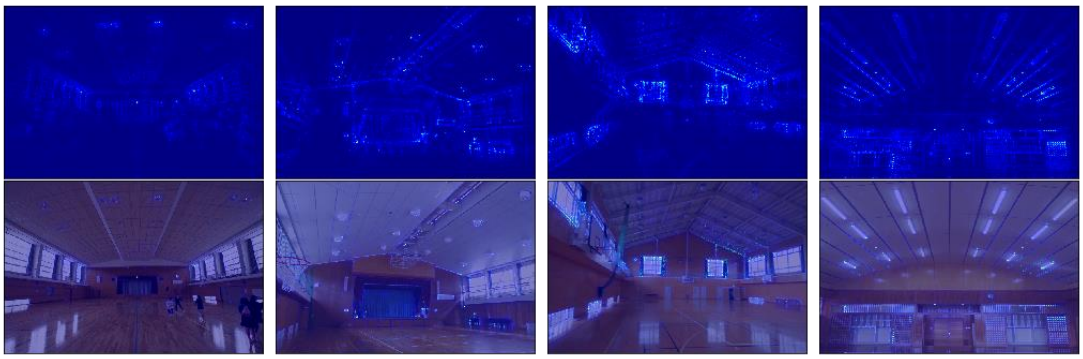


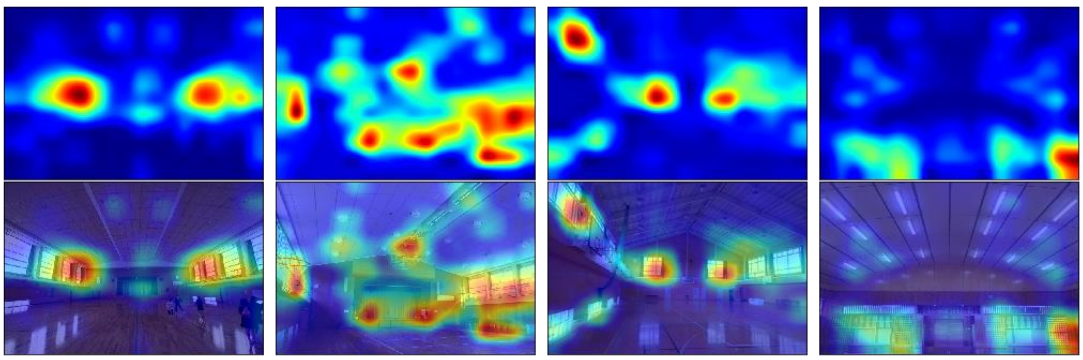


(b)_25 (b)_26 (b)_27 (b)_28

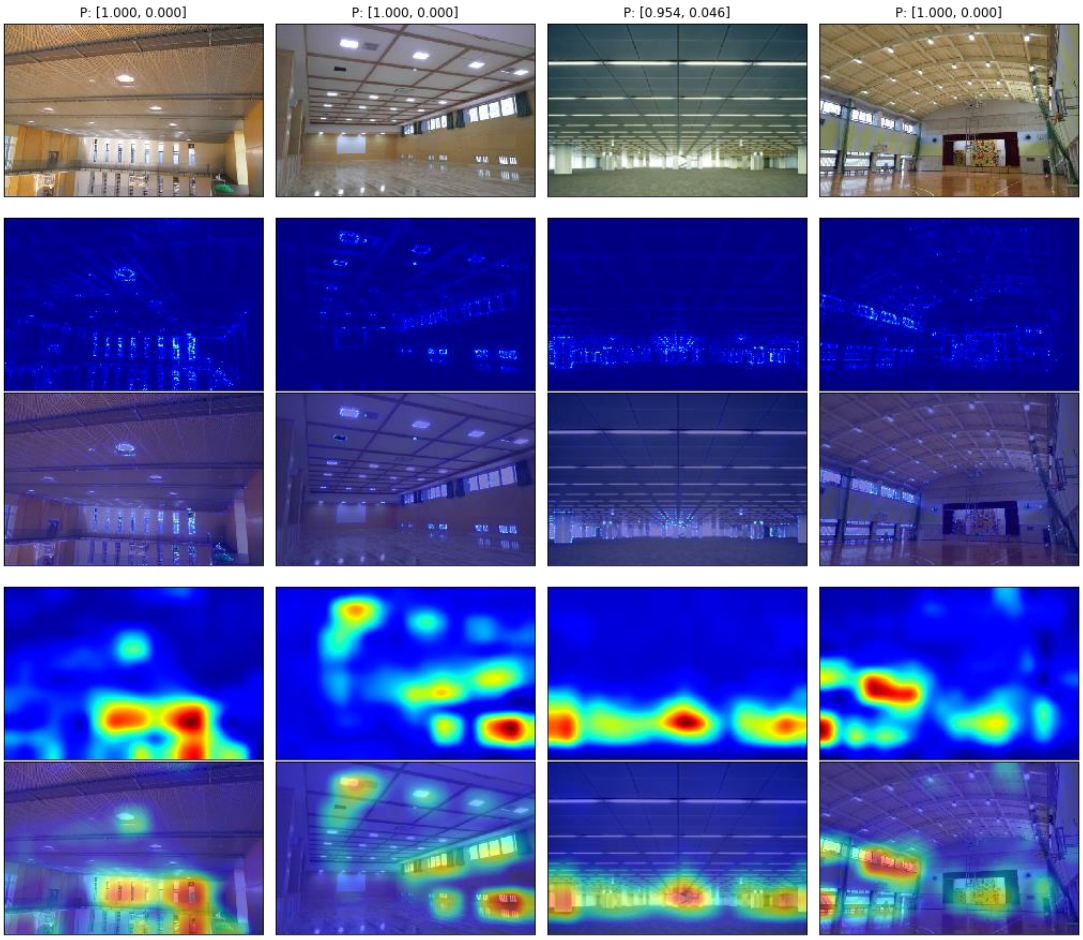
P: [0.997, 0.003] P: [1.000, 0.000] P: [1.000, 0.000] P: [1.000, 0.000]

(b)_29 (b)_30 (b)_31 (b)_32

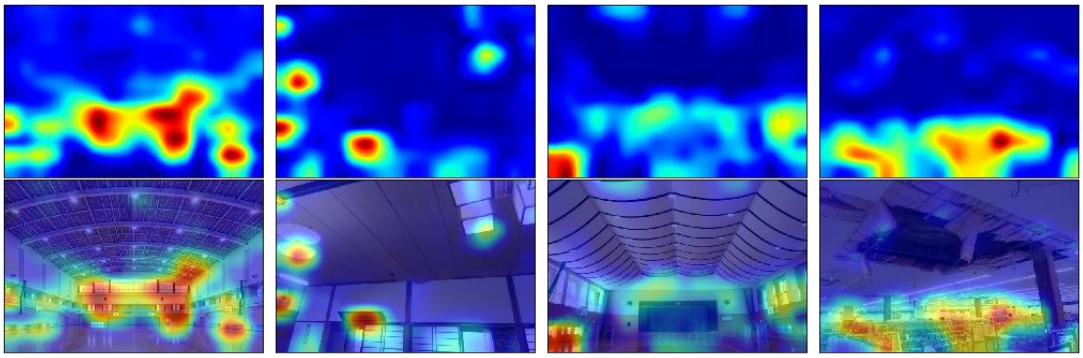




(b)_33 (b)_34 (b)_35 (b)_36



(b)_37 (b)_38 (b)_39 (b)_40



(b)_41

P: [0.358, 0.642]

(b)_42

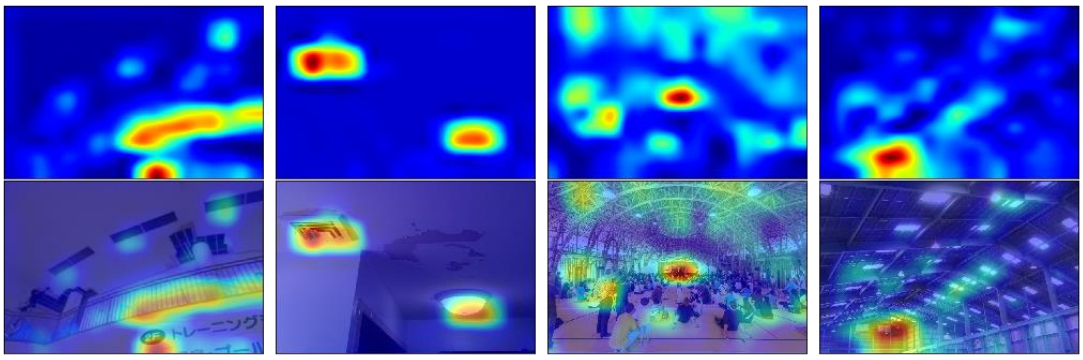
P: [0.106, 0.894]

(b)_43

P: [1.000, 0.000]

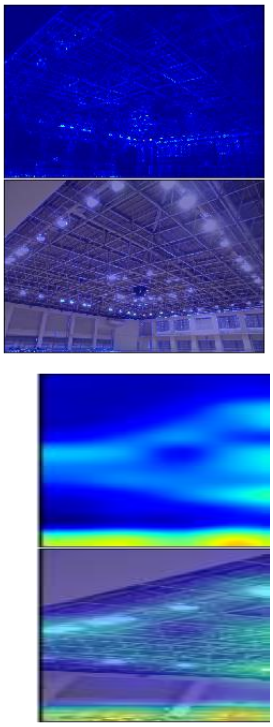
(b)_44

P: [0.913, 0.087]



(b)_45

P: [1.000, 0.000]



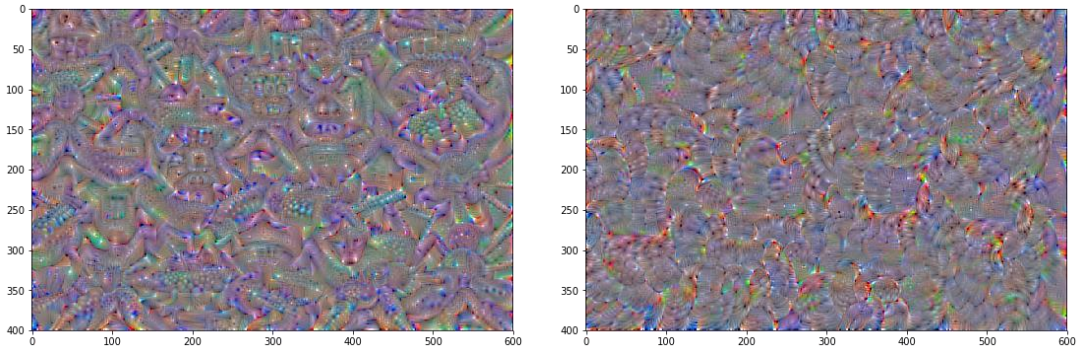
(b) ceiling images from internet

Fig. 5.8 Predictions, saliency maps and Grad-CAM to the images from the testing dataset and internet

5.3.2 The TF_VGG19 model

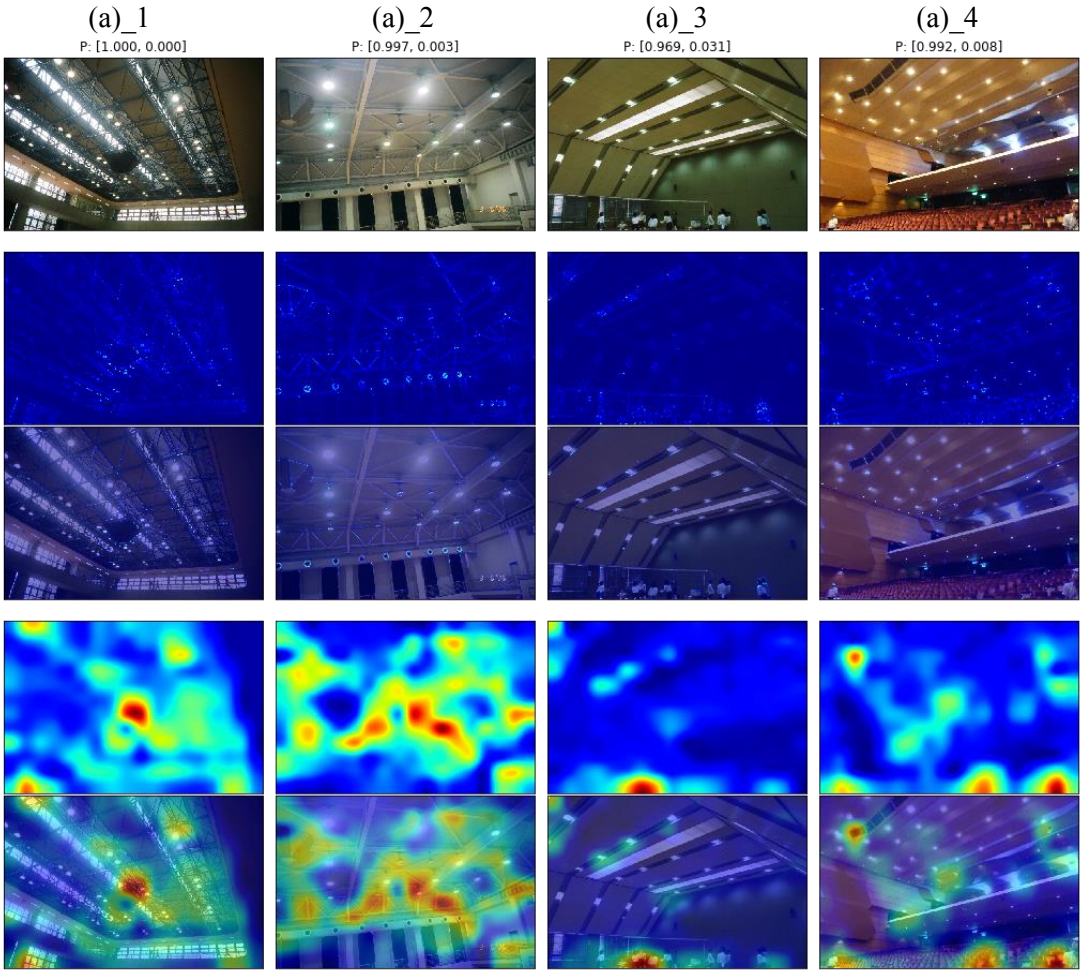
The evaluation of the TF_VGG19 follows the same steps as the TF_VGG16. Firstly, the learnt notions of 'intact' and 'damaged' are visualized in Fig. 5.9. Although they are different from those of the TF_VGG16, the notion images still reflect the characteristic of 'intact' and 'damaged' notions from relating perspectives: Fig. 5.9(a) reflects the roundness of the lights and Fig. 5.9(b) reflects the scattered broken imagery. The final predictions, the saliency maps and the Grad-CAM to the images selected from the testing dataset (a) and the internet (b) are shown in Fig. 5.10.

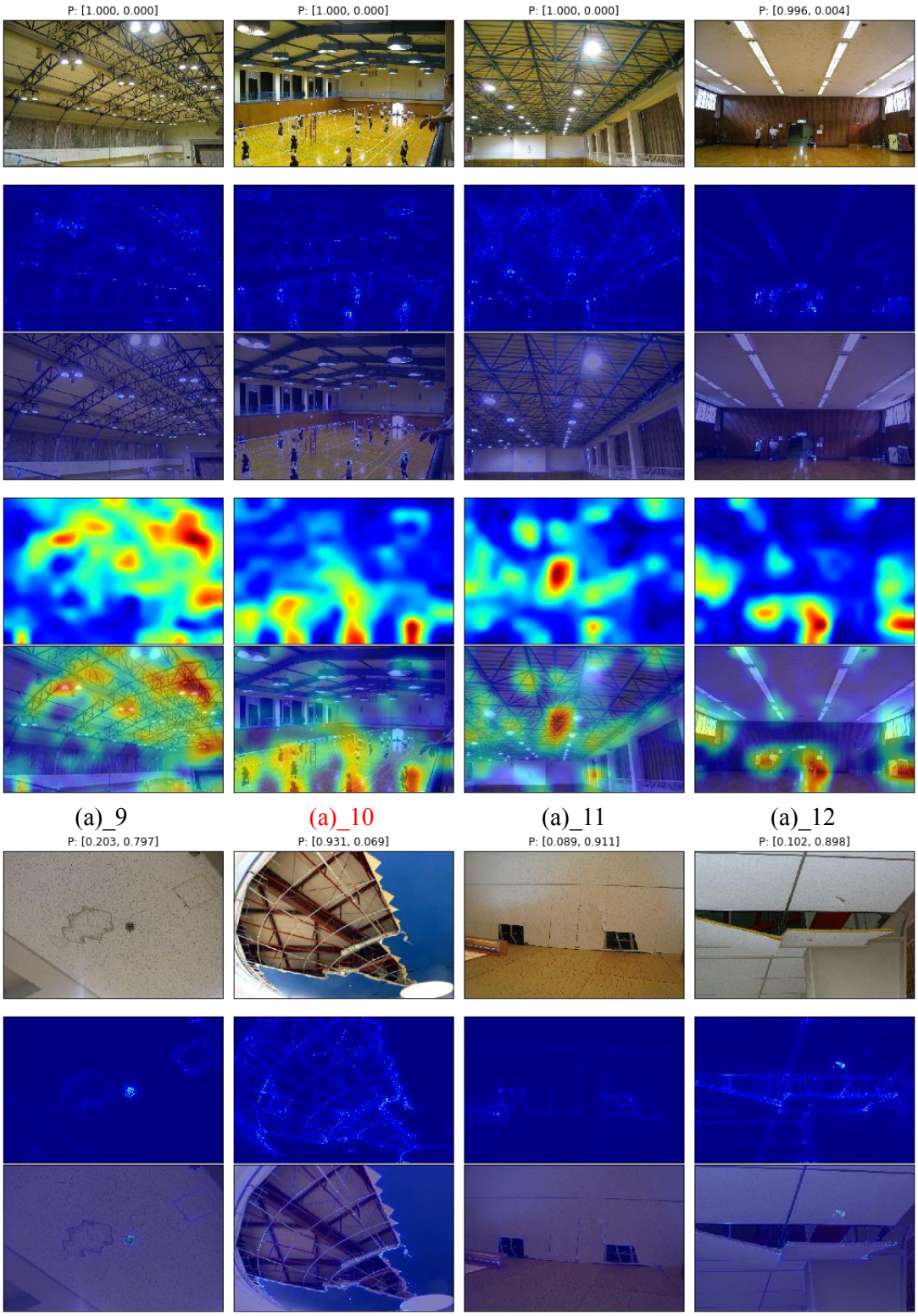
The final prediction accuracy to the testing dataset is 87.5%. The accuracy to the ceiling images from internet is 82.2%, which is lower than that of the TF_VGG16 model and the CNN model in Chapter 3, reflecting that the TF_VGG19 did not learnt enough as expected.

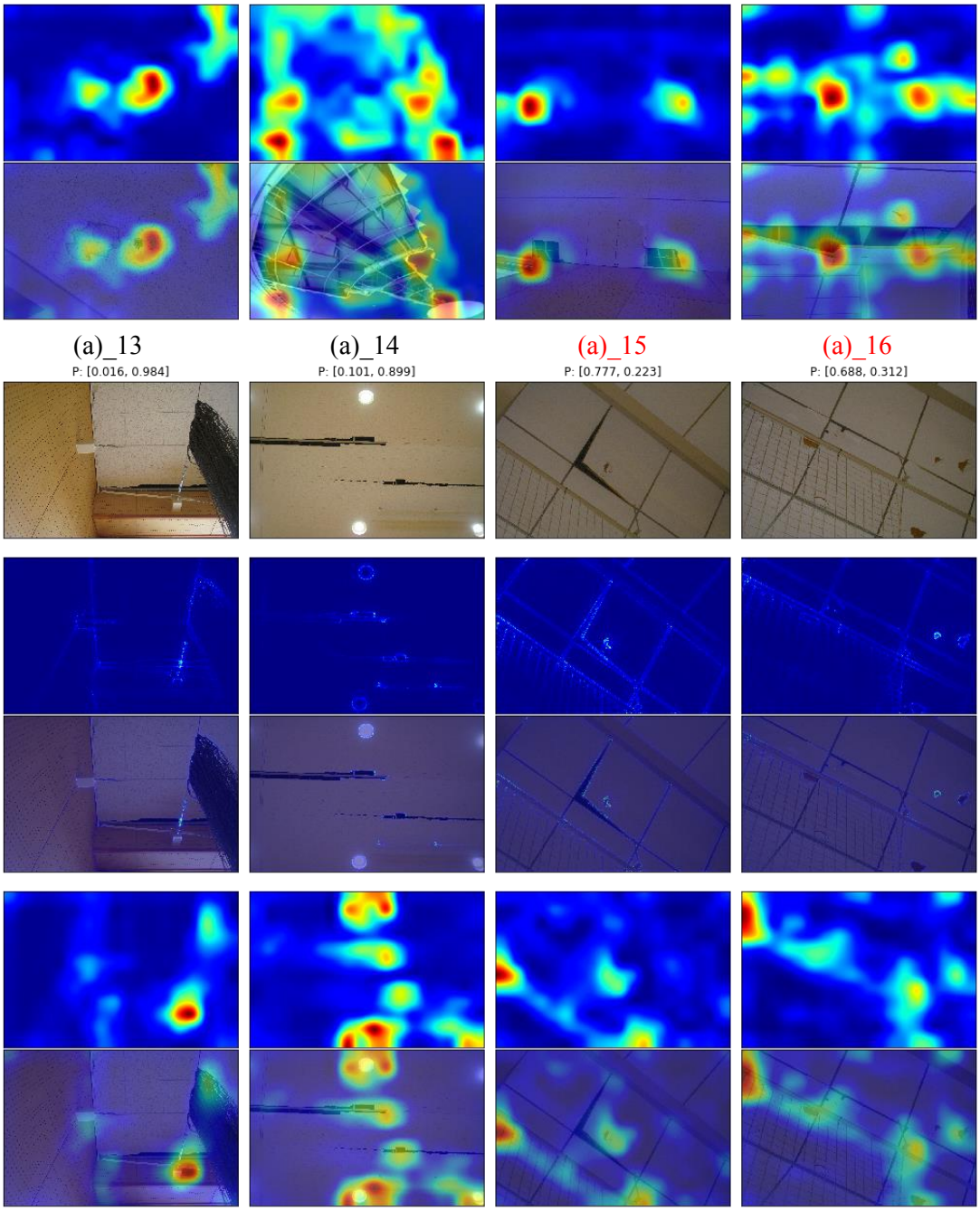


(a) intact (b) damaged

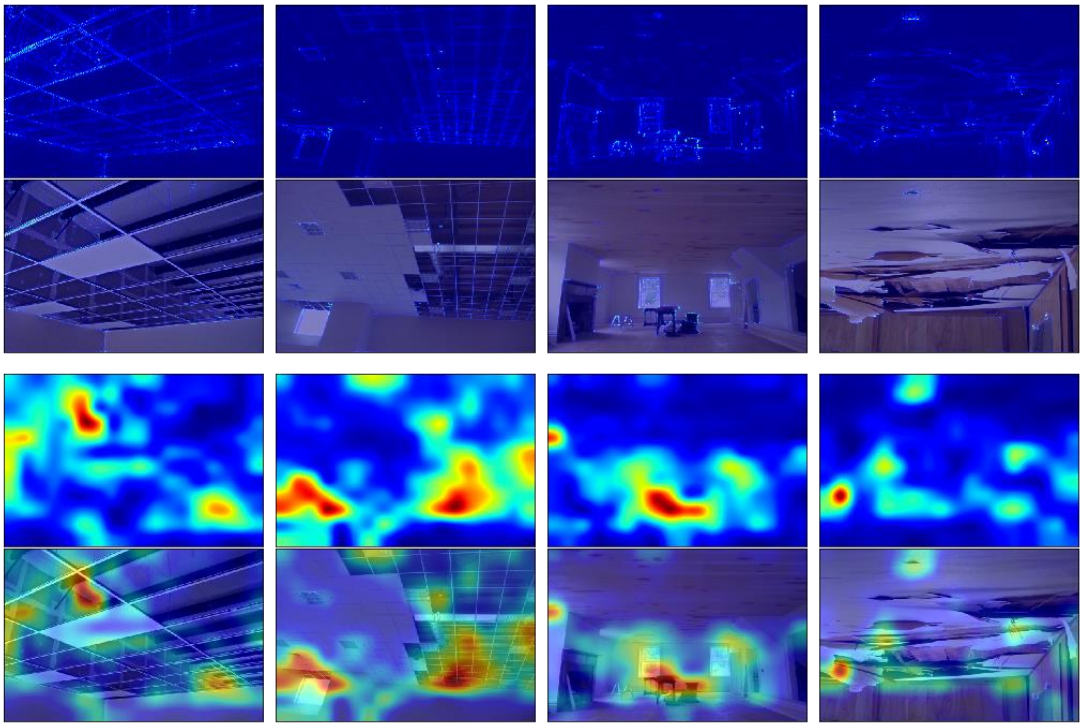
Fig. 5.9 ‘Intact’ and ‘damaged’ images most activating TF_VGG19



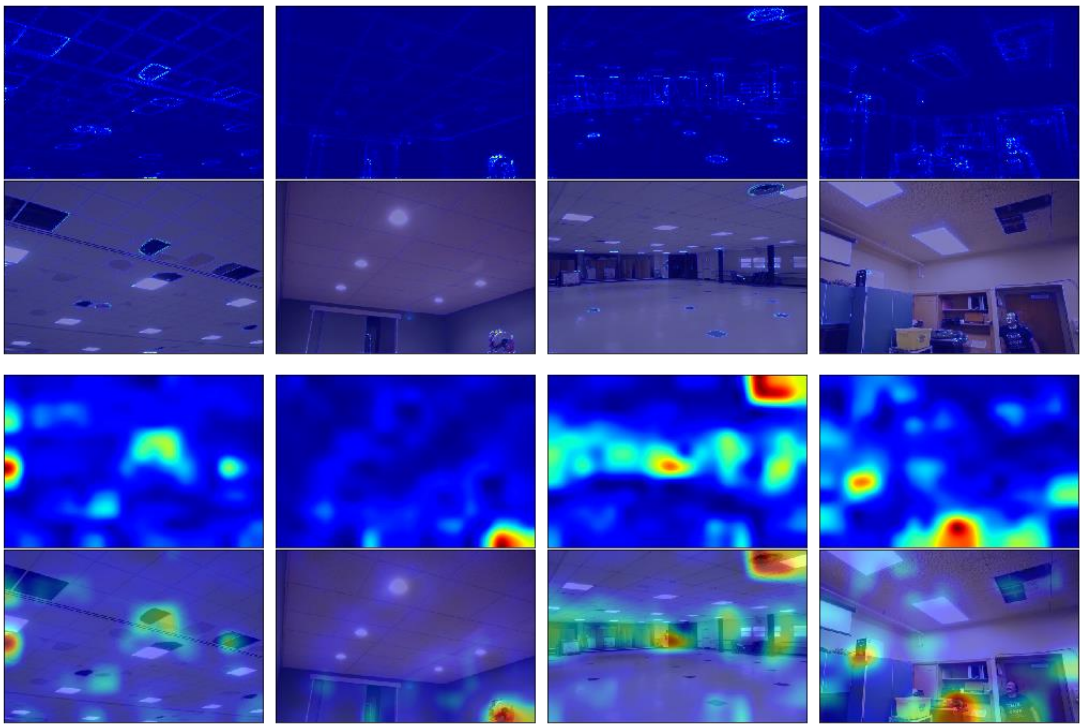

P: [1.000, 0.000]
P: [1.000, 0.000]
P: [1.000, 0.000]
P: [0.996, 0.004]
(a)_9
(a)_10
(a)_11
(a)_12
P: [0.203, 0.797]
P: [0.931, 0.069]
P: [0.089, 0.911]
P: [0.102, 0.898]



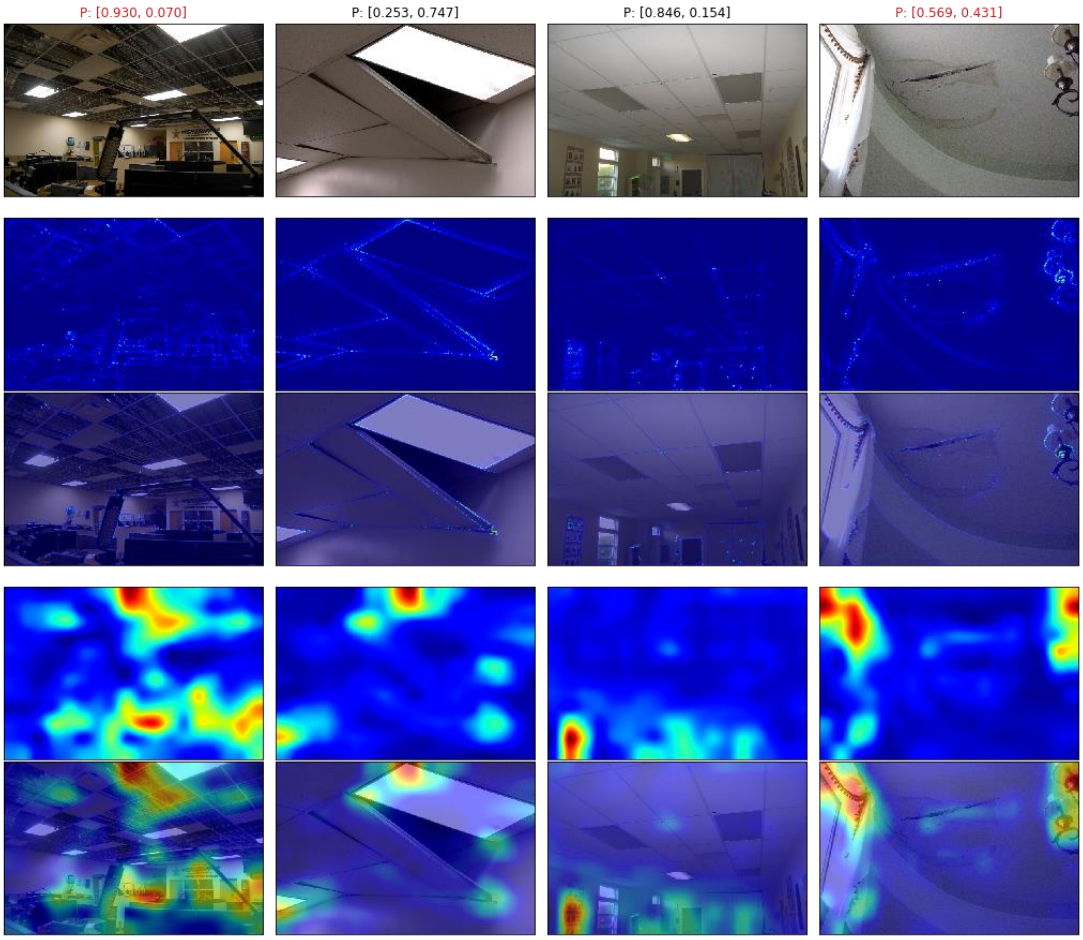
(a) testing dataset



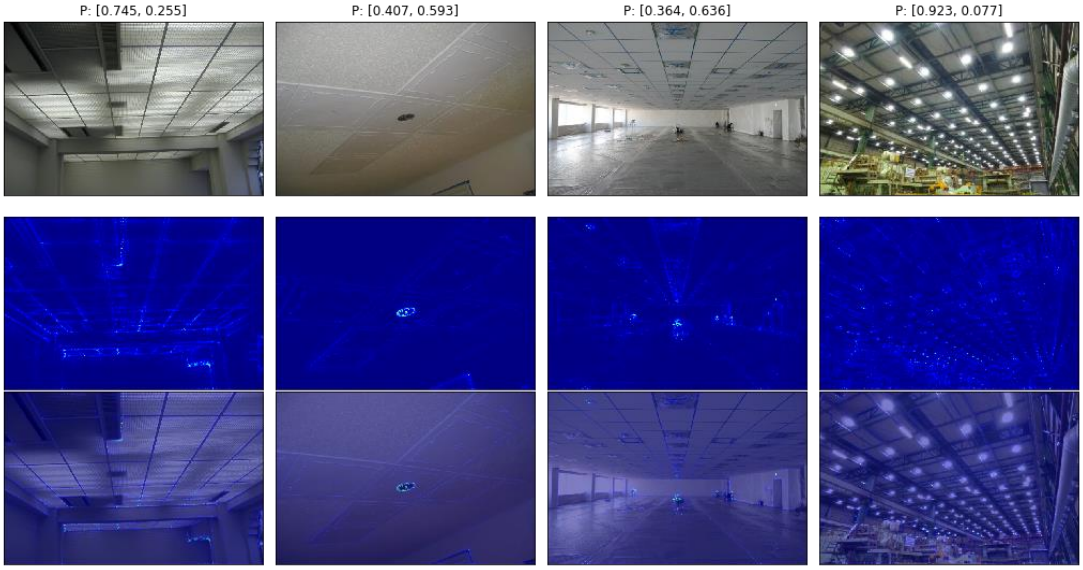
(b)_5 (b)_6 (b)_7 (b)_8

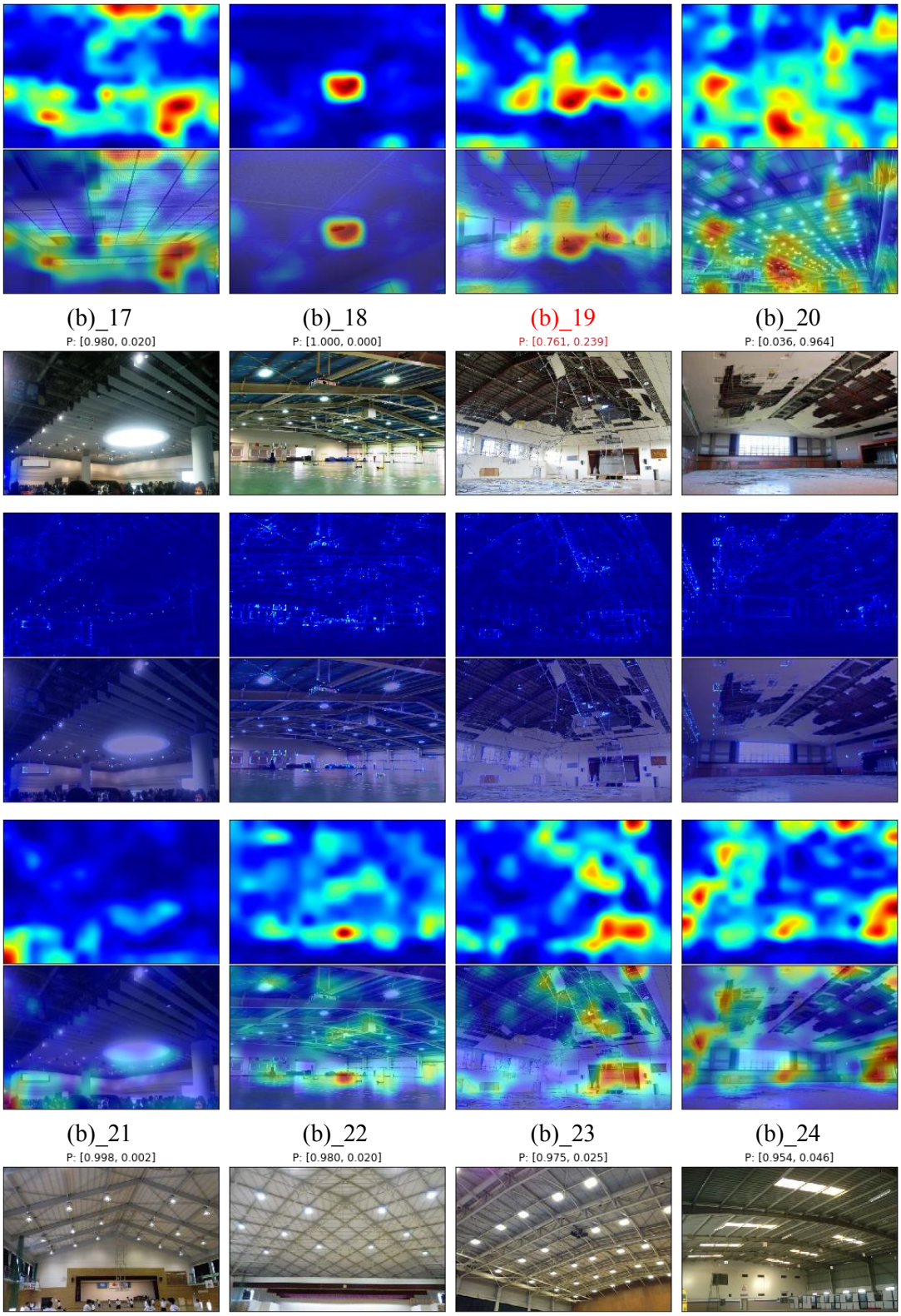


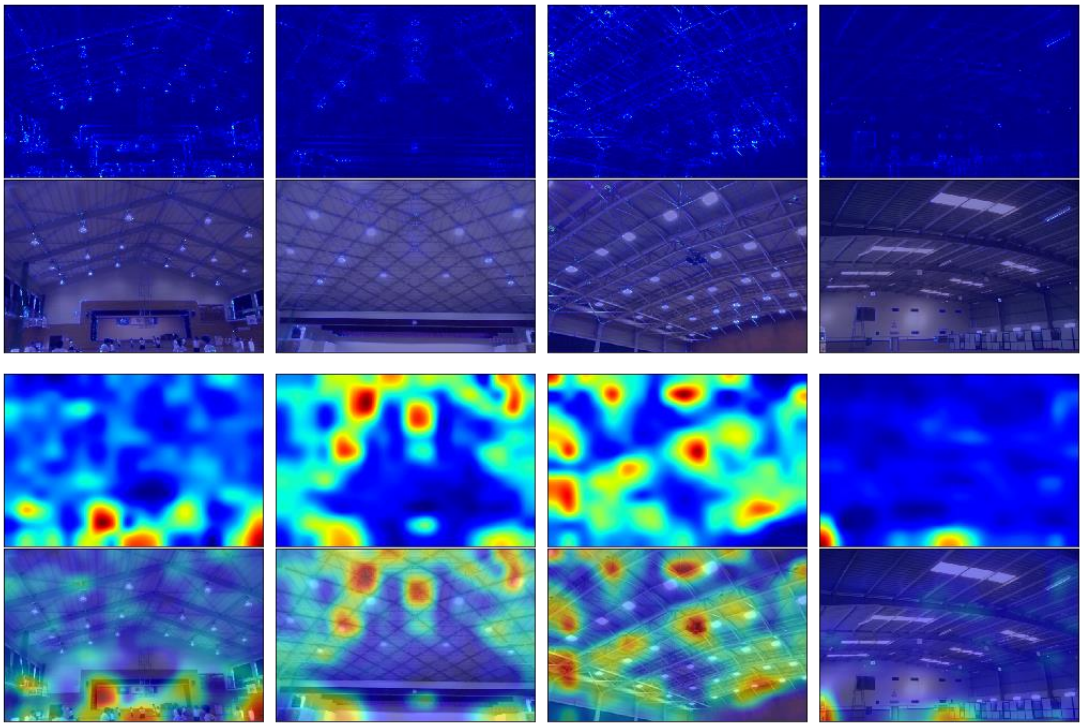
(b)_9 (b)_10 (b)_11 (b)_12



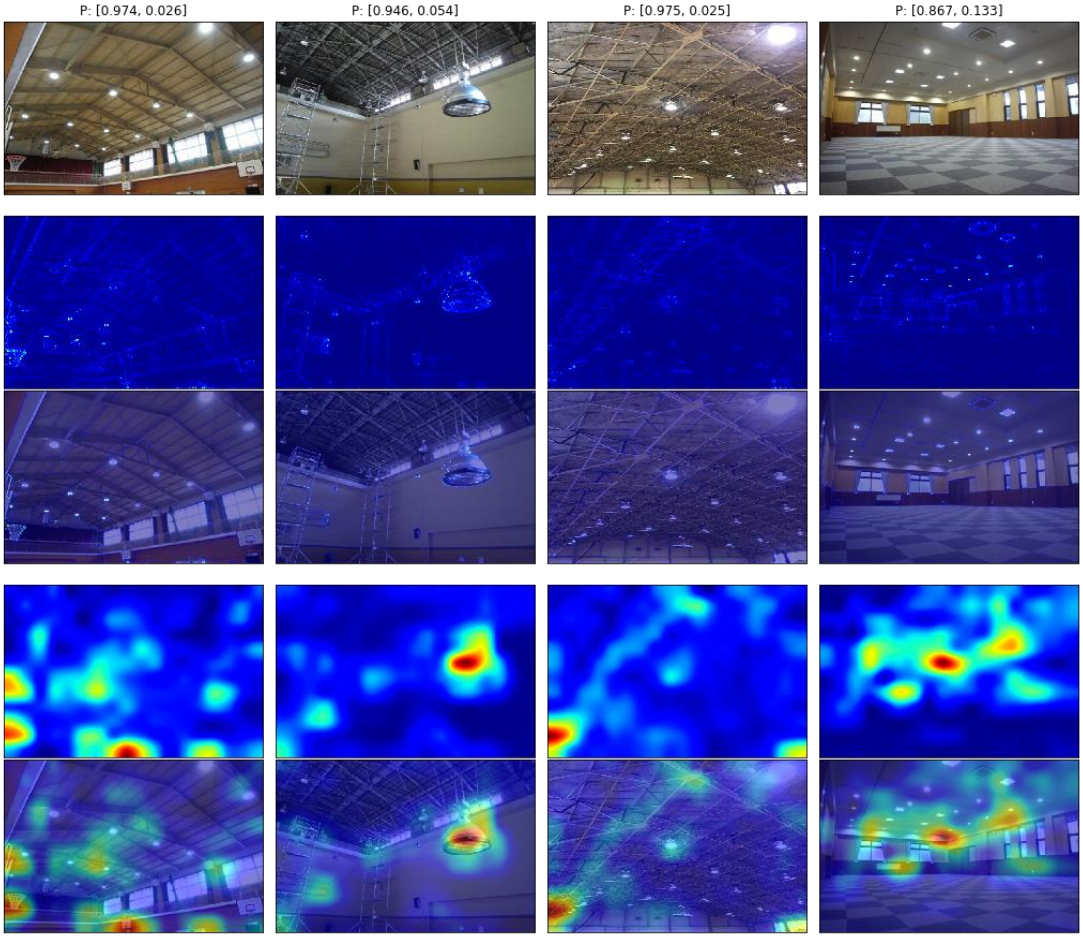
(b)_13 (b)_14 (b)_15 (b)_16



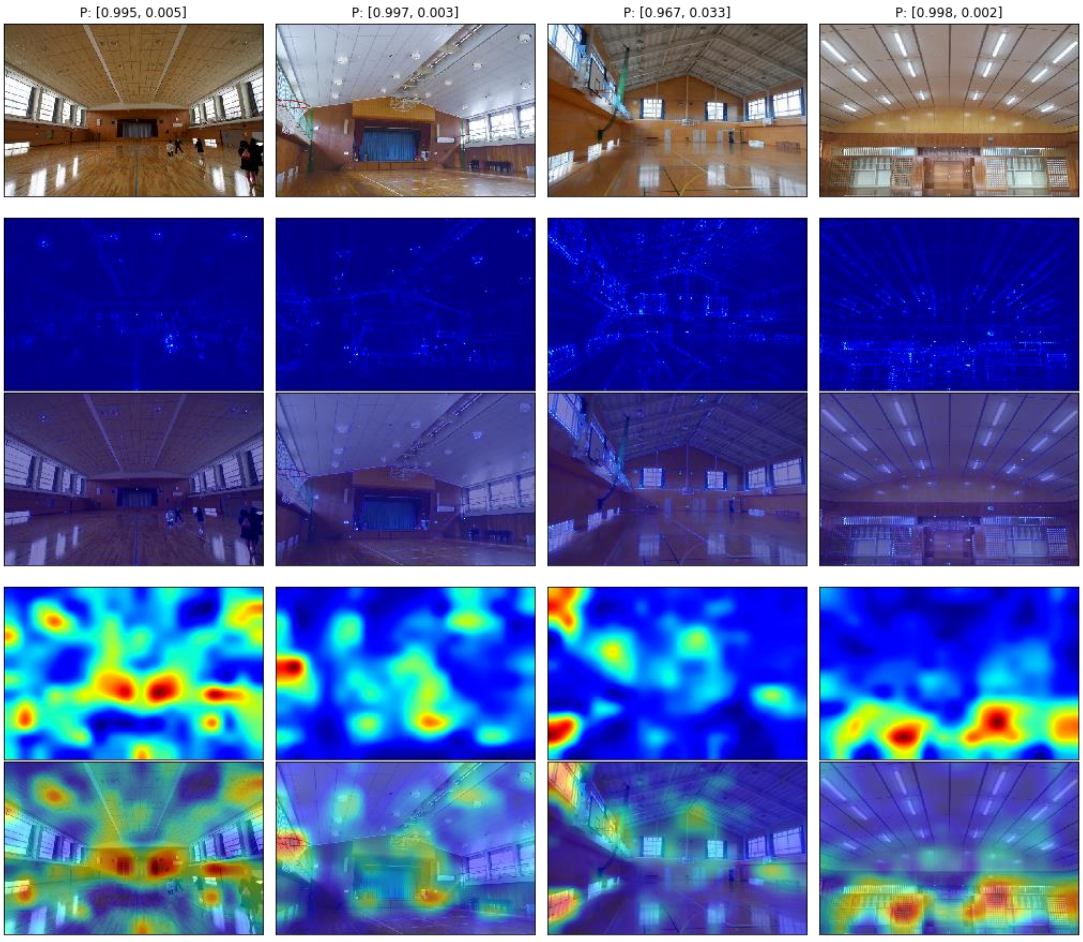

(b)_17
P: [0.980, 0.020]
(b)_18
P: [1.000, 0.000]
(b)_19
P: [0.761, 0.239]
(b)_20
P: [0.036, 0.964]
(b)_21
P: [0.998, 0.002]
(b)_22
P: [0.980, 0.020]
(b)_23
P: [0.975, 0.025]
(b)_24
P: [0.954, 0.046]



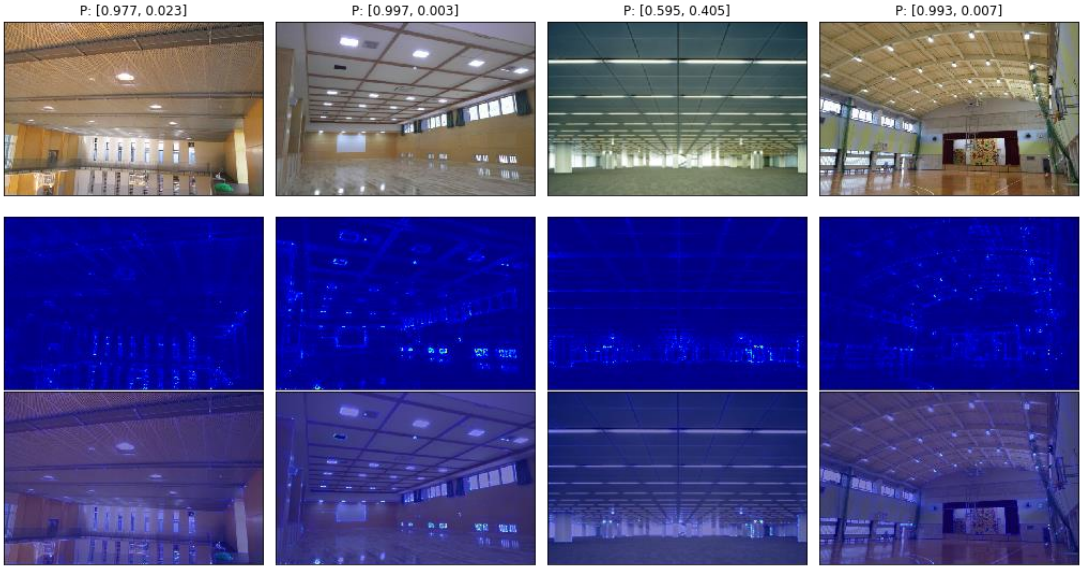
(b)_25 (b)_26 (b)_27 (b)_28

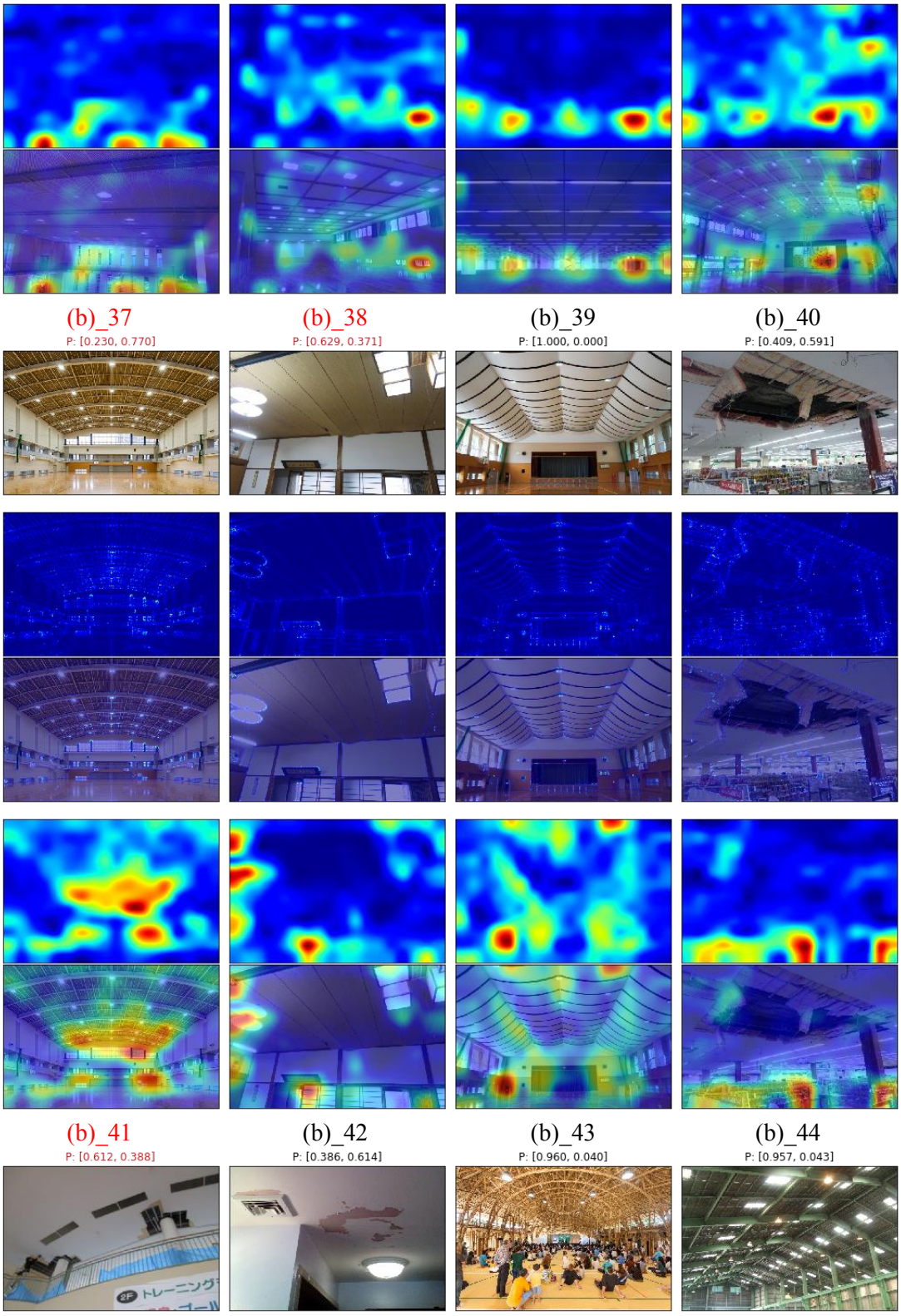


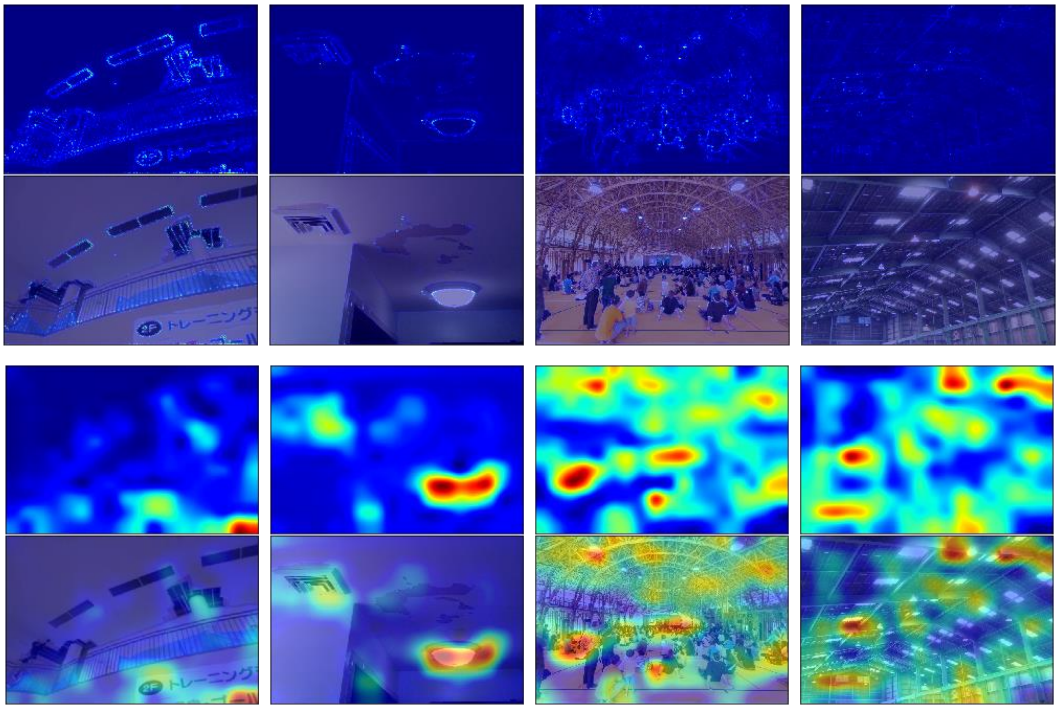
(b)_29 (b)_30 (b)_31 (b)_32



(b)_33 (b)_34 (b)_35 (b)_36




(b)_37
P: [0.230, 0.770]
(b)_38
P: [0.629, 0.371]
(b)_39
P: [1.000, 0.000]
(b)_40
P: [0.409, 0.591]
(b)_41
P: [0.612, 0.388]
(b)_42
P: [0.386, 0.614]
(b)_43
P: [0.960, 0.040]
(b)_44
P: [0.957, 0.043]



(b)_45

(b) ceiling images from internet

Fig. 5.10 Predictions, saliency maps and Grad-CAM to the images from the testing dataset and internet

5.4 Conclusion

In this chapter, two pre-trained CNN models (VGG16 and VGG19) are used as trunk layers for two transfer learning models (TF_VGG16 and TF_VGG19) to perform ceiling damage detection. The conclusions are as follows:

1. The transfer learning models have yielded higher accuracies than that of the CNN model in Chapter 3. The ‘Intact’ and ‘damaged’ images most activating the transfer learning models are much more abstract but interpretable. These indicate that the transfer learning models can recognize more shapes and can learn faster.

2. The saliency maps generated by the transfer learning models have much more details and clarities. This also confirms that the transfer learning models are capable to learn more details from the same training datasets than the CNN model in Chapter 3.

6. Conclusions

A ceiling damage detection system with user-CNN interactive process is proposed in this thesis. Firstly, a CNN model for ceiling damage evaluation is built and trained. Secondly, methods for interpreting and visualizing the predictions made by the CNN model and to the CNN model itself are performed to confirm the reliability of the trained CNN model. The ceiling damage detection function is fulfilled by the saliency map and the Grad-CAM methods. Thirdly, a ceiling damage detection system with user-CNN interactive process is also proposed that the user can be either an expert in ceiling structures who performs routine inspections to the ceilings or a refugee / layman who desperately wants to know if the ceilings over his / her head are safe. This ceiling damage detection system use detection-zoom in-detection repeats that can generate more and more punctual predictions through the interactive process between the user the system. Moreover, the transfer learning method was introduced for ceiling damage detection to build and train more powerful CNN models. The conclusions are:

1. The deep learning (convolutional neural networks) method is possible to be applied in ceiling damage evaluation and detection even if: **a.** only two classes representing 'intact' and 'damaged' (images labeled as the same label can be totally different in manifestation), **b.** lack of training data (approximately only 1000 images for each class); **c.** high resolution of training data (400×600×3, to keep enough information).

2. The cores of this thesis are the interpretations to the trained CNN model. Interpretations include: visualizations of the intermediate convolutional layer outputs, visualizations to activation maps to the filters, and more importantly, visualizations to the saliency map and the Grad-CAM to highlight the pixels contributing most to the final predictions. These visualizations confirm that: **a.** The CNN model performs gradual abstractions through the convolutional layers; **b.** The CNN model has learnt the most representing features from the training data (the 'intact' and the 'damaged' features in the ceilings); **c.** The ceiling damage detection function can be realized by the visualizations of the saliency map and the Grad-CAM methods.

3. A ceiling damage detection system with user-CNN interactive process is raised. The user can either be an expert who inspects the ceilings or a layman who inquires the ceiling working status. It is tested by characteristic damaged ceiling images and interactive performances. The results indicate that the prediction accuracy rises with the damaged regions zoomed in and the process of interactivities between the user and the system. A web-based ceiling damage detection system is possible to collect new ceiling images for further improvement to the CNN model.

4. Transfer learning models using pre-trained VGG16 and VGG19 models as stem layers are proved to be suitable and more efficient in training CNN models for ceiling damage detection. The results indicate that transfer learning models can learn better and faster than the model built in Chapter 3, proved by: **a.** Transfer learning models have higher prediction accuracies; **b.** The saliency maps generated by the transfer learning models are much clearer that those generated by the CNN model built from scratch.

Future work that can build a more powerful ceiling damage detection system include:

1. There are many methods adopt to overcome the scarcity of the original ceiling images, especially the damaged ones. Collecting more representative ceiling images is crucial for the CNN model to be more versatile in ceiling damage evaluation and detection.

2. There are many new deep learning architectures of image recognition in recent years and more of them will emerge in the future. Applying these new architectures to the ceiling damage recognition models may generate more powerful deep learning models.

3. In object recognition, there are also more techniques emerging in recent years. Applying these techniques may generate better damage recognition approaches.

4. The idea of ceiling damage detection by interpreting and visualizing the inner mechanisms of the CNN models is possible to be applied into more domains such as architecture damage detection, medical diagnostic imaging and class-correspondence object detection.

References

1. K. Kawaguchi, et al., *Safety of Interior Spaces of Large Enclosures based on the Damage Investigation of Niigata-Chuetusu and Fukuoka-Seiho-oki Earthquakes (in Japnanese: 新潟中越地震と福岡西方沖地震の被災調査にみる大規模集客施設の内部空間の安全性).* SEISAN KENKYU, 2005.11. **57**(6): p. 39~41.

2. Y. Ogi, et al., *Damage to Non-structural Components in Large Roof Buildings Failed During the Iwate-Miyagi Nairiku Earthquake in 2008 or an Earthquake in the North Shore of Iwate Prefecture in July 24th of 2008 (in Japanese: 平成 20 年（2008 年）岩手・宮城内陸地震または 2008 年 7 月 24 日の岩手県沿岸北部の地震による大規模集客施設の非構造材被害).* AIJ J. Technol., 2010.6. **16**(33): p. 821~826.

3. Y. Ogi, Y. Oba, and K. Kawaguchi, *Damage to Non-structural Components in Large Roof Buildings Failed during the Earthquake in Suruga-Bay of Japan in August 11th of 2009 (in Japanese: 2009 年 8 月 11 日駿河湾の地震による大規模集客施設の非構造材被害).* SEISAN KENKYU, 2009.10. **61**(6): p. 1035~1041.

4. 国土交通省国土技術政策総合研究所, 独立行政法人建築研究所, *2003 年十勝沖地震における空港ターミナルビル等の天井の被害に関する現地調査報告.* 2003.10. p. 24.

5. K. Kawaguchi, et al., *Failure of Suspended Ceilings in Large Public Spaces by Great East Japan Earthquake (in Japanese: 東日本大震災における公共大空間施設での天井落下被害事例).* SEISAN KENKYU, 2011.11. **63**(6): p. 63~70.

6. H. Tomioka, et al., *Study on Seismic Performance of Conventional Type Ceiling : Part 1~Part4 (in Japanese: 在来天井の耐震性に関する研究 その 1~その 4).* Summaries of technical papers of annual meeting B-1, 2013.7: p. 1103~1101.

7. 国土交通省国土技術政策総合研究所, 独立行政法人建築研究所, *豊田スタジアムスポーツプラザ屋内プールの天井板脱落の現地調査報告.* 2008.1.

8. R. Hosomi, et al., *Preliminary Investigation on the Non-seismic failure of Non*

Structural Components occurred at Fuji-Shi on July 15, 2013 (in Japanese: 非地震時の天井落下事例（2013 年 7 月 15 日富士市）に関する基礎的調査研究). Summaries of technical papers of annual meeting 2015 B-1, 2014.7: p. 909～910.

9. 国土交通省国土技術政策総合研究所, 独立行政法人建築研究所, *スポパーク松森における天井落下事故調査報告 −大空間を有するスポーツ等施設の天井落下−*. 2005.8. p. 27.

10. T. Uchida, et al., *Preliminary study of safety of non-structural components in wide roof buildings : Part 1: Floor area and suspended height of ceiling (in Japanese: 大規模集客施設内部の非構造材に関する基礎的調査研究 : その 1:フロア面積と天井設置高さに関するアンケート調査)*. Summaries of technical papers of Annual Meeting Architectural Institute of Japan A-1, 2008.7: p. 209~210.

11. S. Sakurai, et al., *Fundamental study of non-seismic failure of suspended ceilings of swimming pools : Pull out strength of screws in humid circumstances (in Japanese: 非地震時における屋内プール天井の落下被害に関する基礎的考察 : 吸水時のビスの頭抜け強度について)*. Summaries of technical papers of annual meeting Architectural Institute of Japan B-1, 2009.7: p. 897~898.

12. AIJ, *Guidelines for Safety Measures Against Accidental Fall of Ceilings and Other Non-structural Components (in Japanese: 天井等の非構造材の落下に対する安全対策指針・同解説)*. 2015: AIJ.

13. S. Katayama, et al., *Preliminary study of safety of non-structural components in wide roof buildings : Part 3: Impact hammer experiment with a dummy head (in Japanese: 大規模集客施設内部の非構造材に関する基礎的調査研究 : その 3:インパクトハンマーによる人頭模型の応答実験)*. Summaries of technical papers of Annual Meeting Architectural Institute of Japan. A-1, 2008.7: p. 213~214.

14. Y. Nakaso, et al., *Fundamental Research on the Safety Criteria of Nonstructural Components in Large Enclosures: Drop Tests Using Plaster Boards (in Japanese: 天井材の安全性評価に関する基礎的研究：石膏ボード落下実験)*. SEISAN KENKYU, 2012.11. **64**(6): p. 95~100.

15. Y. Nakaso and K. Kawaguchi, *Fundamental research on the safety criteria of nonstructural components in large enclosures using human tolerance index :*

Part 6: Estimation of the ceiling drop impact based on the transfer function (in Japanese: 人体耐性指標を用いた天井材の安全性評価に関する基礎的研究 その 6 伝達関数に基づく天井材落下衝撃荷重の推定). Summaries of technical papers of annual meeting B-1, 2013.7: p. 1007~1008.

16. Y. Nakaso and K. Kawaguchi, *Fundamental research on the safety criteria of nonstructural components in large enclosures using human tolerance index : Part 7: Identification of the optimal transfer function (in Japanese: 人体耐性指標を用いた天井材の安全性評価に関する基礎的研究 その 7 最適伝達関数の同定)*. Summaries of technical papers of annual meeting B-1, 2014.7: p. 921~922.

17. K. Kawaguchi, Y. Oba, and Y. Nakaso, *Failure of Ceilings and Estimation of Its Impact Occurred in an Airport Building During the 2011 Off the Pacific Coast of Tohoku Earthquake (in Japanese: 2011 年東北地方太平洋沖地震による空港ターミナルビル内天井落下及び天井落下衝撃力の推定)*. AIJ J. Technol., 2012.6. **39**(18): p. 789~793.

18. Y. Nakaso, *Identification of Impact Load of Dropped Ceilings to a Dummy Head by Inverse Analysis (in Japanese: 天井落下時に発生する頭部衝撃荷重の逆問題解析による同定に関する研究)*, in *Dept. of Architecture, School of Engineering, The Univ. of Tokyo*. 2015.2, The Univ. of Tokyo.

19. I. Maruyama, et al., *CARBONATION AND FASTENING STRENGTH DETERIORATION OF CALCIUM SILICATE BOARD:-Analysis of materials collected from a ceiling collapse accident (in Japanese: けい酸カルシウム板の中性化と留付け強度低下に関する研究 ： ー天井落下事故を生じた材料の分析ー)*. J. Struct. Constr. Eng., 2013.7. **78**(689): p. 1203~1208.

20. K. Kawaguchi, *Report on large roof structures damaged by the Great Hanshin-Awaji earthquake.* International Journal of Space Structures, 1997. **12**(3-4): p. 137-147.

21. K. Kawaguchi. *Damage to non-structural components in large rooms by the Japan earthquake*. in *Structures Congress 2012, March 29, 2012 - March 31, 2012*. 2012. Chicago, IL, United states: American Society of Civil Engineers (ASCE).

22. Ministry of Education, Sports, Science and Technology - Japan. *Guidance for Ceiling Falling Prevention in School Facilities (In Japanese: 学校施設にお*

ける天井等落下防止対策のための手引). 2013 Aug; Available from: http://www.nier.go.jp/shisetsu/pdf/ceiling_all.pdf.

23. Ministry of Education, Sports, Science and Technology - Japan. *Case Studies of Countermeasures to Ceiling Falling in Indoor Sports Halls (In Japanese: 屋内運動場等の天井等落下防止対策事例集)*. 2014 April; Available from: http://www.nier.go.jp/shisetsu/pdf/anti-drop_all.pdf.

24. Ministry of Education, Sports, Science and Technology - Japan. *Guidebook for Earthquake Protection for Nonstructural Members of School Facilities (Revised Edition) Protecting Children from Falling and Tumbling Objects due to an Earthquake – Implementing Earthquake Resistance Inspection –*. 2015 March; Available from: http://www.nier.go.jp/shisetsu/pdf/e-gijyutsu2.pdf.

25. Ministry of Land, Transport and Tourism - Japan. *Notification No. 282 of the Ministry of Land, Infrastructure, Transport and Tourism (In Japanese: 建築物の定期調査報告における調査及び定期点検における点検の項目、方法及び結果の判定基準並びに調査結果表を定める件)*. 2008 March; Available from: http://wwwkt.mlit.go.jp/notice/pdf/201703/00006549.pdf.

26. C. Sodeikat and F. Knab, *Aufnahme von historischen Deckensystemen mit verschiedenen Methoden der zerstörungsfreien Prüfung ZfP.* Beton- Und Stahlbetonbau, 2014. **109**(7): p. 453-462.

27. A. Pereira, et al., *Inspection and diagnosis system for gypsum plasters in partition walls and ceilings.* Construction and Building Materials, 2011. **25**(4): p. 2146-2156.

28. Y. Nitta, et al., *Development of the damage assessment methodology for ceiling elements*, in *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012, Pts 1 and 2*, M. Tomizuka, C.B. Yun, and J.P. Lynch, Editors. 2012, Spie-Int Soc Optical Engineering: Bellingham.

29. C. Cadena, et al., *Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age.* IEEE Transactions on Robotics, 2016. **32**(6): p. 1309-1332.

30. S.-Y. Hwang and J.-B. Song, *Clustering and probabilistic matching of arbitrarily shaped ceiling features for monocular vision-based SLAM.* Advanced Robotics, 2013. **27**(10): p. 739-747.

31. M. Jung and J.-B. Song, *Robust mapping and localization in indoor environments.* Intelligent Service Robotics, 2016. **10**(1): p. 55-66.

32. S.W. Doebling, et al., *Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review.* 1996.

33. C.C. Ciang, J.-R. Lee, and H.-J. Bang, *Structural health monitoring for a wind turbine system: a review of damage detection methods.* Measurement Science and Technology, 2008. **19**(12): p. 122001.

34. J. Ko and Y. Ni, *Technology developments in structural health monitoring of large-scale bridges.* Engineering structures, 2005. **27**(12): p. 1715-1725.

35. E.P. Carden and P. Fanning, *Vibration Based Condition Monitoring: A Review.* Structural Health Monitoring, 2004. **3**(4): p. 355-377.

36. Y. Zou, L. Tong, and G.P. Steven, *Vibration-based model-dependent damage (delamination) identification and health monitoring for composite structures—a review.* Journal of Sound and Vibration, 2000. **230**(2): p. 357-378.

37. O. Salawu, *Detection of structural damage through changes in frequency: a review.* Engineering structures, 1997. **19**(9): p. 718-723.

38. Y. Lu and J.E. Michaels, *A methodology for structural health monitoring with diffuse ultrasonic waves in the presence of temperature variations.* Ultrasonics, 2005. **43**(9): p. 717-731.

39. S. Liang, et al., *Fiber-optic intrinsic distributed acoustic emission sensor for large structure health monitoring.* Optics Letters, 2009. **34**(12): p. 1858-1860.

40. D. Broda, et al., *Modelling of nonlinear crack–wave interactions for damage detection based on ultrasound—A review.* Journal of Sound and Vibration, 2014. **333**(4): p. 1097-1118.

41. R. Curadelli, et al., *Damage detection by means of structural damping identification.* Engineering structures, 2008. **30**(12): p. 3497-3504.

42. A.-M. Yan, et al., *Structural damage diagnosis under varying environmental conditions—part II: local PCA for non-linear cases.* Mechanical Systems and Signal Processing, 2005. **19**(4): p. 865-880.

43. A.-M. Yan, et al., *Structural damage diagnosis under varying environmental conditions—part I: a linear analysis.* Mechanical Systems and Signal Processing, 2005. **19**(4): p. 847-864.

44. L. Mujica, et al., *Q-statistic and T2-statistic PCA-based measures for damage assessment in structures.* Structural Health Monitoring, 2011. **10**(5): p. 539-553.

45. J.-H. Chou and J. Ghaboussi, *Genetic algorithm in structural damage detection.* Computers & Structures, 2001. **79**(14): p. 1335-1353.

46. R. Perera and R. Torres, *Structural damage detection via modal data with genetic algorithms.* Journal of Structural Engineering, 2006. **132**(9): p. 1491-1501.

47. H.Z. HosseinAbadi, et al., *GUW-based structural damage detection using WPT statistical features and multiclass SVM.* Applied Acoustics, 2014. **86**: p. 59-70.

48. X. Wu, J. Ghaboussi, and J. Garrett Jr, *Use of neural networks in detection of structural damage.* Computers & Structures, 1992. **42**(4): p. 649-659.

49. P. Pandey and S. Barai, *Multilayer perceptron in damage detection of bridge structures.* Computers & Structures, 1995. **54**(4): p. 597-608.

50. C. Zang and M. Imregun, *Structural damage detection using artificial neural networks and measured FRF data reduced via principal component projection.* Journal of Sound and Vibration, 2001. **242**(5): p. 813-827.

51. L. Yam, Y. Yan, and J. Jiang, *Vibration-based damage detection for composite structures using wavelet transform and neural network identification.* Composite Structures, 2003. **60**(4): p. 403-412.

52. X. Fang, H. Luo, and J. Tang, *Structural damage detection using neural network with learning rate improvement.* Computers & Structures, 2005. **83**(25-26): p. 2150-2161.

53. D. Ziou and S. Tabbone, *Edge detection techniques-an overview.* Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii, 1998. **8**: p. 537-559.

54. I. Abdel-Qader, O. Abudayyeh, and M.E. Kelly, *Analysis of Edge-Detection Techniques for Crack Identification in Bridges.* Journal of Computing in Civil

Engineering, 2003. **17**(4): p. 255-263.

55. T. Nishikawa, et al., *Concrete Crack Detection by Multiple Sequential Image Filtering.* Computer-Aided Civil and Infrastructure Engineering, 2012. **27**(1): p. 29-47.

56. M. O'Byrne, et al., *Texture Analysis Based Damage Detection of Ageing Infrastructural Elements.* Computer-Aided Civil and Infrastructure Engineering, 2013. **28**(3): p. 162-177.

57. J.B. Butcher, et al., *Defect Detection in Reinforced Concrete Using Random Neural Architectures.* Computer-Aided Civil and Infrastructure Engineering, 2014. **29**(3): p. 191-207.

58. Y. LeCun, et al., *Convolutional Networks and Applications in Vision*, in *2010 Ieee International Symposium on Circuits and Systems*. 2010, Ieee: New York. p. 253-256.

59. Y.-J. Cha, W. Choi, and O. Büyüköztürk, *Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks.* Computer-Aided Civil and Infrastructure Engineering, 2017. **32**(5): p. 361-378.

60. D. Soukup and R. Huber-Mork, *Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images*, in *Advances in Visual Computing*, G. Bebis, et al., Editors. 2014, Springer-Verlag Berlin: Berlin. p. 668-677.

61. Y.z. Lin, Z.h. Nie, and H.w. Ma, *Structural Damage Detection with Automatic Feature‐Extraction through Deep Learning.* Computer‐Aided Civil and Infrastructure Engineering, 2017. **32**(12): p. 1025-1046.

62. Y.J. Cha, et al., *Autonomous Structural Visual Inspection Using Region‐Based Deep Learning for Detecting Multiple Damage Types.* Computer‐Aided Civil and Infrastructure Engineering, 2017.

63. R.C. Gonzalez and R.E. Woods, *Digital Image Processing (3rd Edition)*. 2006: Prentice-Hall, Inc.

64. J. Canny, *A computational approach to edge detection.* Ieee Transactions on Pattern Analysis and Machine Intelligence, 1986(6): p. 679-698.

65. D.H. Ballard, *Generalizing the Hough transform to detect arbitrary shapes.* Pattern Recognition, 1981. **13**(2): p. 111-122.

66. T.M. Mitchell, *Machine Learning*. 1997: McGraw-Hill, Inc. 432.

67. Y.S. Abu-Mostafa, M. Magdon-Ismail, and H.T. Lin, *Learning from Data: A Short Course*. 2012: AMLBook.com.

68. E. Alpaydin, *Introduction to Machine Learning, 3rd Edition*. Introduction to Machine Learning, 3rd Edition. 2014, Cambridge: Mit Press. 1-613.

69. D. Silver, et al., *Mastering the game of Go with deep neural networks and tree search.* Nature, 2016. **529**(7587): p. 484-489.

70. D. Silver, et al., *Mastering the game of go without human knowledge.* Nature, 2017. **550**(7676): p. 354.

71. O. Russakovsky, et al., *Imagenet large scale visual recognition challenge.* International Journal of Computer Vision, 2015. **115**(3): p. 211-252.

72. D. Castelvecchi, *Can we open the black box of AI?* Nature News, 2016. **538**(7623): p. 20.

73. G. Ian, B. Yoshua, and C. Aaron, *Deep Learning*. 2016: MIT Press.

74. W.S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 1943. **5**(4): p. 115-133.

75. D.O. Hebb, *The organization of behavior: A neuropsychological theory*. 1949, New York: Wiley.

76. Y. LeCun, *Generalization and network design strategies.* Connectionism in perspective, 1989: p. 143-155.

77. D. Williams and G. Hinton, *Learning representations by back-propagating errors.* Nature, 1986. **323**(6088): p. 533-538.

78. G.E. Hinton. *Learning distributed representations of concepts*. in *Proceedings of the eighth annual conference of the cognitive science society*. 1986. Amherst, MA.

79. B. Schölkopf, C.J. Burges, and A.J. Smola, *Advances in kernel methods: support vector learning*. 1999: MIT press.

80. M.I. Jordan, *Learning in graphical models*. Vol. 89. 1998: Springer Science & Business Media.

81. C. Cortes and V. Vapnik, *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.

82. B.E. Boser, I.M. Guyon, and V.N. Vapnik. *A training algorithm for optimal margin classifiers*. in *Proceedings of the fifth annual workshop on Computational learning theory*. 1992. ACM.

83. G.E. Hinton and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks.* Science, 2006. **313**(5786): p. 504-507.

84. C. Poultney, S. Chopra, and Y.L. Cun. *Efficient learning of sparse representations with an energy-based model*. in *Advances in neural information processing systems*. 2007.

85. Y. Bengio, et al. *Greedy layer-wise training of deep networks*. in *Advances in neural information processing systems*. 2007.

86. R. Pascanu, et al., *How to construct deep recurrent neural networks.* arXiv preprint arXiv:1312.6026, 2013.

87. G.F. Montufar, et al. *On the number of linear regions of deep neural networks*. in *Advances in neural information processing systems*. 2014.

88. Y. Bengio and Y. LeCun, *Scaling learning algorithms towards AI.* Large-scale kernel machines, 2007. **34**(5): p. 1-41.

89. Y. Bengio, *Learning Deep Architectures for AI.* Foundations and Trends® in Machine Learning, 2009. **2**(1): p. 1-127.

90. Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning.* Nature, 2015. **521**(7553): p. 436-444.

91. M. Ahmadlou and H. Adeli, *Enhanced probabilistic neural network with local decision circles: A robust classifier.* Integrated Computer-Aided Engineering, 2010. **17**(3): p. 197-210.

92. Z. Cui, et al., *Deep Network Cascade for Image Super-resolution*, in *Computer Vision - Eccv 2014, Pt V*, D. Fleet, et al., Editors. 2014, Springer Int Publishing Ag: Cham. p. 49-64.

93. A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*. 2009.

94. A. Krizhevsky, I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.

95. R. Girshick. *Fast R-CNN*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

96. R. Girshick, et al., *Rich feature hierarchies for accurate object detection and semantic segmentation*, in *2014 Ieee Conference on Computer Vision and Pattern Recognition*. 2014, Ieee: New York. p. 580-587.

97. K. He, et al., *Mask R-CNN.* arXiv preprint arXiv:1703.06870, 2017.

98. A. Karpathy and L. Fei-Fei, *Deep Visual-Semantic Alignments for Generating Image Descriptions.* IEEE Trans Pattern Anal Mach Intell, 2017. **39**(4): p. 664-676.

99. T.Y. Lin, et al., *Microsoft COCO: Common Objects in Context*, in *Computer Vision - Eccv 2014, Pt V*, D. Fleet, et al., Editors. 2014, Springer Int Publishing Ag: Cham. p. 740-755.

100. H. Noh, S. Hong, and B. Han. *Learning deconvolution network for semantic segmentation*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

101. S. Ren, et al., *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* IEEE Trans Pattern Anal Mach Intell, 2017. **39**(6): p. 1137-1149.

102. E. Shelhamer, J. Long, and T. Darrell, *Fully Convolutional Networks for Semantic Segmentation.* IEEE Trans Pattern Anal Mach Intell, 2017. **39**(4): p. 640-651.

103. I.J. Goodfellow, et al., *Multi-digit number recognition from street view imagery*

using deep convolutional neural networks. arXiv preprint arXiv:1312.6082, 2013.

104. G. Hinton, et al., *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.* IEEE Signal Processing Magazine, 2012. **29**(6): p. 82-97.

105. M. Abadi, et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems.* arXiv preprint arXiv:1603.04467, 2016.

106. F. Bastien, et al., *Theano: new features and speed improvements.* arXiv preprint arXiv:1211.5590, 2012.

107. R. Collobert, K. Kavukcuoglu, and C. Farabet. *Torch7: A matlab-like environment for machine learning*. in *BigLearn, NIPS Workshop*. 2011.

108. Y. Jia, et al. *Caffe: Convolutional architecture for fast feature embedding*. in *Proceedings of the 22nd ACM international conference on Multimedia*. 2014. ACM.

109. F. Chollet, *Keras*. 2015.

110. A. Esteva, et al., *Dermatologist-level classification of skin cancer with deep neural networks.* Nature, 2017. **542**(7639): p. 115-118.

111. G.E. Dahl, N. Jaitly, and R. Salakhutdinov, *Multi-task neural networks for QSAR predictions.* arXiv preprint arXiv:1406.1231, 2014.

112. F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.* Psychological review, 1958. **65**(6): p. 386.

113. Y. LeCun, et al., *Efficient backprop.* Neural Networks: Tricks of the Trade, 1998. **1524**: p. 9-50.

114. M. Blaauw and J. Bonada, *A Neural Parametric Singing Synthesizer.* arXiv preprint arXiv:1704.03809, 2017.

115. A.v.d. Oord, et al., *Wavenet: A generative model for raw audio.* arXiv preprint arXiv:1609.03499, 2016.

116. V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning.*

arXiv preprint arXiv:1603.07285, 2016.

117. Y. LeCun and Ieee, *Deep Learning & Convolutional Networks.* 2015 Ieee Hot Chips 27 Symposium (Hcs), 2016: p. 121.

118. C. Szegedy, et al., *Going Deeper with Convolutions*, in *2015 Ieee Conference on Computer Vision and Pattern Recognition*. 2015, Ieee: New York. p. 1-9.

119. C. Zhang, et al., *Understanding deep learning requires rethinking generalization.* arXiv preprint arXiv:1611.03530, 2016.

120. M.D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, in *Computer Vision - Eccv 2014, Pt I*, D. Fleet, et al., Editors. 2014, Springer Int Publishing Ag: Cham. p. 818-833.

121. Y. LeCun, et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.

122. M. Bojarski, et al., *End to end learning for self-driving cars.* arXiv preprint arXiv:1604.07316, 2016.

123. N. Srivastava, et al., *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research, 2014. **15**: p. 1929-1958.

124. A.G. Lalkhen and A. McCluskey, *Clinical tests: sensitivity and specificity.* Continuing Education in Anaesthesia Critical Care & Pain, 2008. **8**(6): p. 221-223.

125. M.D. Zeiler, G.W. Taylor, and R. Fergus. *Adaptive deconvolutional networks for mid and high level feature learning*. in *2011 International Conference on Computer Vision*. 2011.

126. B. Zhou, et al., *Object detectors emerge in deep scene cnns.* arXiv preprint arXiv:1412.6856, 2014.

127. G.E. Hinton, S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets.* Neural computation, 2006. **18**(7): p. 1527-1554.

128. D. Erhan, et al., *Visualizing higher-layer features of a deep network.* University of Montreal, 2009. **1341**(3): p. 1.

129. Q.V. Le. *Building high-level features using large scale unsupervised learning*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013. IEEE.

130. K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps.* arXiv preprint arXiv:1312.6034, 2013.

131. M. Alexander, O. Christopher, and T. Mike. *Inceptionism: Going Deeper into Neural Networks*. 2015; Available from: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

132. E.L. Spratt, *Dream Formulations and Deep Neural Networks: Humanistic Themes in the Iconology of the Machine-Learned Image.* arXiv preprint arXiv:1802.01274, 2018.

133. J. Yosinski, et al., *Understanding neural networks through deep visualization.* arXiv preprint arXiv:1506.06579, 2015.

134. K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.

135. F. Wang, et al., *Residual attention network for image classification.* arXiv preprint arXiv:1704.06904, 2017.

136. H. Zheng, et al. *Learning multi-attention convolutional neural network for fine-grained image recognition*. in *Int. Conf. on Computer Vision*. 2017.

137. A. Das, et al., *Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions?* Computer Vision and Image Understanding, 2017. **163**: p. 90-100.

138. J.T. Springenberg, et al., *Striving for simplicity: The all convolutional net.* arXiv preprint arXiv:1412.6806, 2014.

139. M. Lin, Q. Chen, and S. Yan, *Network in network.* arXiv preprint arXiv:1312.4400, 2013.

140. B. Zhou, et al. *Learning deep features for discriminative localization*. in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*.

2016. IEEE.

141. R.R. Selvaraju, et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.* See https://arxiv.org/abs/1610.02391 v3, 2016.

142. S. Thrun and L. Pratt, *Learning to learn*. 2012: Springer Science & Business Media.

143. K. Weiss, T.M. Khoshgoftaar, and D. Wang, *A survey of transfer learning.* Journal of Big Data, 2016. **3**(1): p. 9.

144. P.H. Calais Guerra, et al. *From bias to opinion: a transfer-learning approach to real-time sentiment analysis*. in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011. ACM.

145. Y. Zhu, et al. *Heterogeneous Transfer Learning for Image Classification*. in *AAAI*. 2011.

146. S.J. Pan and Q. Yang, *A survey on transfer learning.* IEEE Transactions on knowledge and data engineering, 2010. **22**(10): p. 1345-1359.

147. J. Blitzer, et al. *Learning bounds for domain adaptation*. in *Advances in neural information processing systems*. 2008.

148. S. Ben-David, et al. *Analysis of representations for domain adaptation*. in *Advances in neural information processing systems*. 2007.

149. S. Ben-David, et al., *A theory of learning from different domains.* Machine learning, 2010. **79**(1-2): p. 151-175.

150. J. Yosinski, et al. *How transferable are features in deep neural networks?* in *Advances in neural information processing systems*. 2014.

Appendix A: URLs of the images from internet

URLs of the images from internet:

1. https://tupolecam.pl/sufity-podwieszane/konstrukcja-stelaz/
2. https://paslaugos.lt/vaidas-vv385/galerija/376749
3.http://festivalsalsacali.com/the-outrageous-real-drop-ceiling-ideas-for-basement-ideas/ideas-of-awesome-basement-ceiling-ideas-3-most-popular-basement-ceiling-that-awesome/
4. https://usercontent2.hubstatic.com/4966769_f1024.jpg
5.http://www.servprocassstjosephcounties.com/FranchiseContent/GalleryPhotos/8714-319c7366-f021-40c9-948e-c8d244c8ba83.JPG
6.http://crapimissedit.com/i/2018/02/best-paint-for-ceiling-tiles-how-to-paint-acoustic-ceiling-tiles-ceiling-paint-spray-can-how-to-repair-ceiling-tiles-with-water-damage.jpg
7. http://www.troop125ny.org/eagle-nest2/JamesRyan/eg1.jpg
8.http://www.dailyastorian.com/storyimage/DA/20161018/ARTICLE/161019724/EP/1/1/EP-161019724.jpg&MaxW=600
9. https://flaglerlive.com/wp-content/uploads/eoc-tiles.jpg
10.http://ktva.images.worldnow.com/images/14560502_G.jpg?auto=webp&disable=upscale&height=560&fit=bounds
11. https://cupertinopianogames.files.wordpress.com/2011/12/img_2587.jpg
12.http://peckdrywallandpainting.com/wp-content/uploads/2013/03/Water-Damaged-Skip-Trowel-Ceiling-Melbourne.jpg
13. http://www.emono1.jp/img/kamiyar/20120422223554_img1_5.jpg
14.https://crapimissedit.com/i/2018/02/how-to-paint-acoustic-ceiling-tiles-best-paint-for-ceiling-tiles-how-to-repair-ceiling-tiles-with-water-damage-zinsser-covers-up-ceiling-paint-970x728.jpg
15. http://www.met-s.co.jp/images/material/original_img001.jpg
16. http://www.e-lmx.com/showcase.html#
17. http://www.perle-st.co.jp/display_ceiling.html
18. http://www.leadray.com/comm/upimage/p_151025_04834.jpg
19. http://blogimg.goo.ne.jp/user_image/38/ee/1cc79efd68b2523e6463ee131e2dde98.jpg
20.http://safty.sakura.ne.jp/sblo_files/safty-living/image/E4BD93E882B2E9A4A8E5A4A9E4BA95E890BDE4B88BE58699E79C9F001.jpg
21. http://www.schoolnews.jp/wp-content/uploads/2015/06/2df269c7fd43000e02d79d1e88188612.jpg
22. http://www.schoolnews.jp/wp-content/uploads/2015/06/8b23b0ee5d5bee38ffd161e861b30519.jpg
23. http://www.toyoda-gosei.co.jp/upload/news/436/fcec62f373571adc6f330539dcbb78d9.jpg
24. http://pyramidgroup.in/wp-content/uploads/sites/1/nggallery/industrial/pyramid-industrial6.jpg
25.https://www.osaka-c.ed.jp/blog/semboku-y/katou/images/271109%20%E4%BD%93%E8%82%B2%E9%A4%A8%E7%85%A7%E6%98%8E%E6%94%B9%E4%BF%AE%E3%83%BC%EF%BC%91%20IMG_2217.jpg
26.http://www.honmoku-ac-seikosha.com/news/entry_file_display.php?Name=151ed019529321452d1ffee77c9b5743c1cd695c.jpg&ID=c086dc97f533b829e6ac027d6d2ffd61341e1847
27. http://www.healthy-clay.com/wp/wp-content/uploads/59d58d211a8487eec3879f47e92ec00c.jpg
28. http://sports-mura.com/images_sisetsu/65_4.jpg
29. http://www.oiler.co.jp/kensou/img/000002_list_image.jpg
30. http://www.schoolnews.jp/wp-content/uploads/2015/12/8742dffddaf2dd5ae365ab7642b4f608.jpg
31. http://www.emono1.jp/img/toei-japan/20160906204043_image_9.jpg
32. http://www.kobayashi-denko.co.jp/_p/730/images/pc/9f617c30.JPG
33. http://www.sakcs.jp/wpcms/wp-content/uploads/2014/09/2bc9745e8c83d4d0dac6b74176efee03.jpg
34. http://www.kobayashi-denko.co.jp/_p/730/images/pc/1be11dbd.JPG
35. http://www.toprise.co.jp/products/img/03system/system10.jpg
36. http://www.kumamoto-ymca.or.jp/rifuresu/file/36178.jpg
37. http://livedoor.blogimg.jp/vsnpnet/imgs/e/7/e7379b3a.jpg
38. http://www.konkokyo.or.jp/kakudan/osakarescue/wp-content/uploads/2016/04/S_11288587.jpg

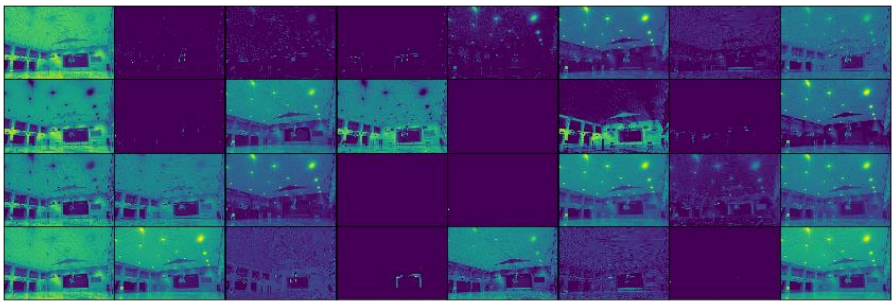
39. http://www.business-directory.jp/image/service1/3/6404.jpg
40. http://livedoor.blogimg.jp/tigerhouse/imgs/d/b/dbf97761.jpg
41. http://jyukanrisystem.com/wp-content/uploads/2017/01/075.jpg
42. https://goguiltypleasures.files.wordpress.com/2013/11/house-fail-ceiling.jpg
43.https://images.adsttc.com/media/images/5987/b9da/b22e/3883/6a00/00b9/slideshow/Bamboo_Sports_Hall_Panyaden_School_(7).jpg?1502067141
44. http://www.another-day.co.jp/blog/DSCF3840.JPG
45. http://www.koyou-m.co.jp/column/img/img164kd4103_1.jpg

Appendix B: Details of the intermediate outputs by the CNN model

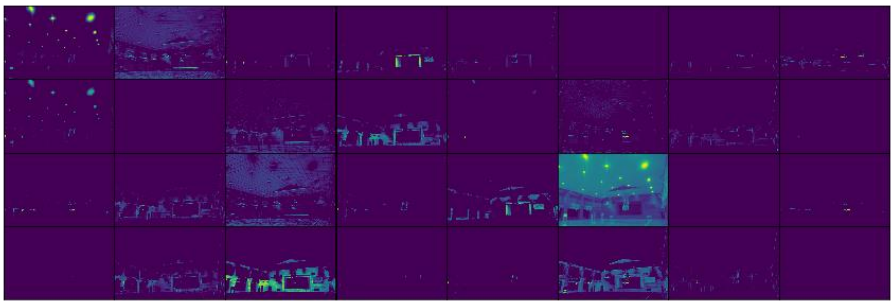
Details of the intermediate outputs by the CNN model:

Table B1 Outputs of the convolutional layers

Input image (Fig. 4.1 (a)), prediction: 0.006

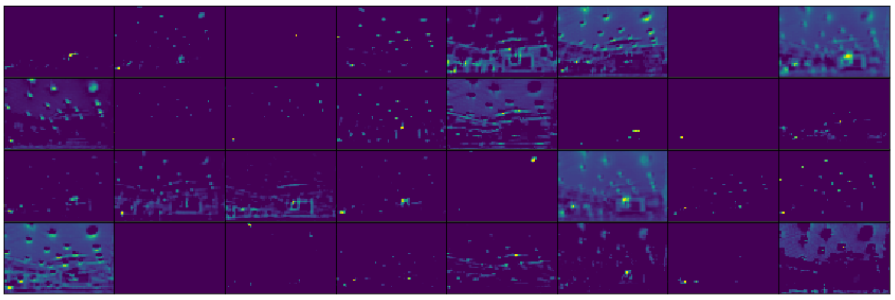


CL1 output: (399, 599, 32)

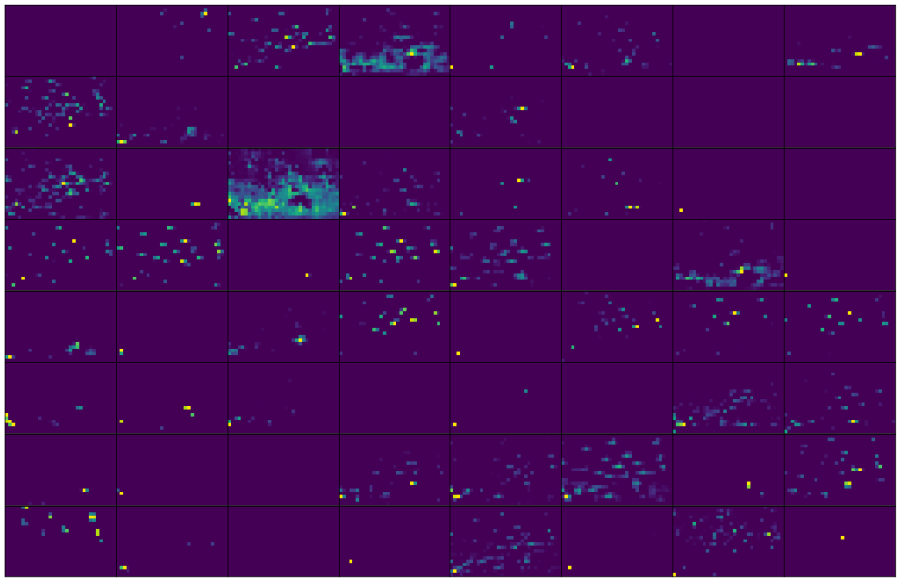


CL4 output: (198, 298, 32)

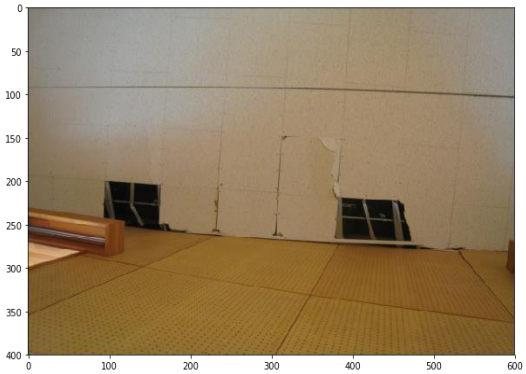
CL7 output: (97, 147, 32)



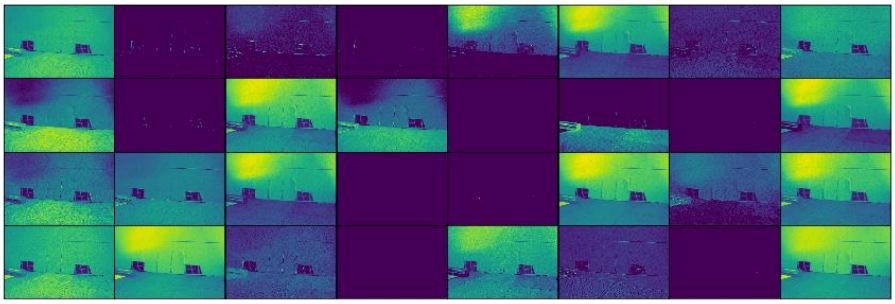
CL10 output: (46, 71, 32)



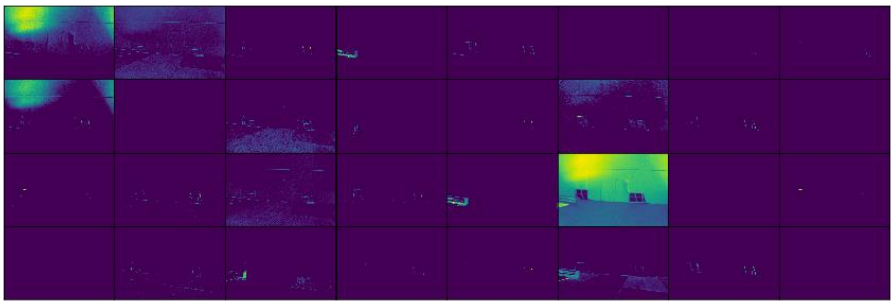
CL13 output: (21, 33, 64)



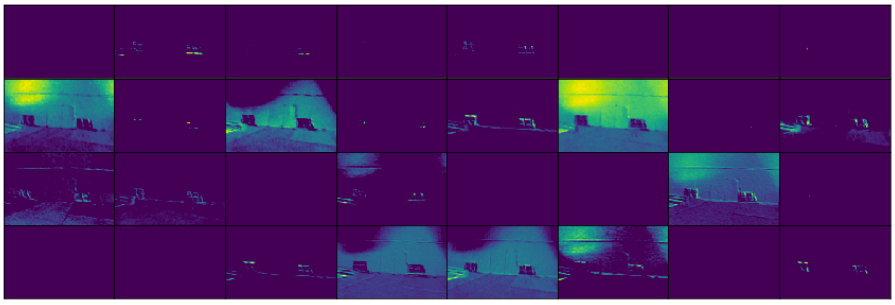
Input image (Fig. 4.1 (b)), prediction: 0.673



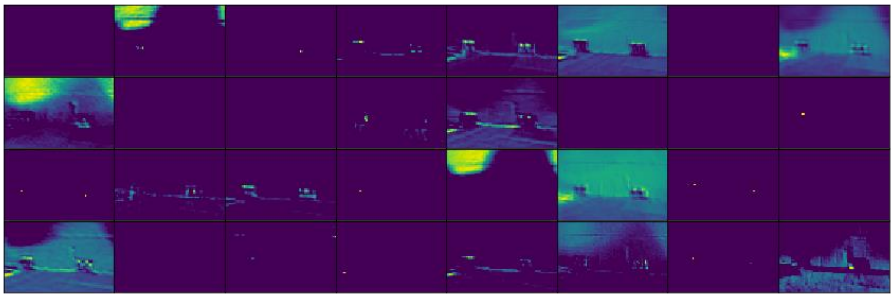
CL1 output: (399, 599, 32)



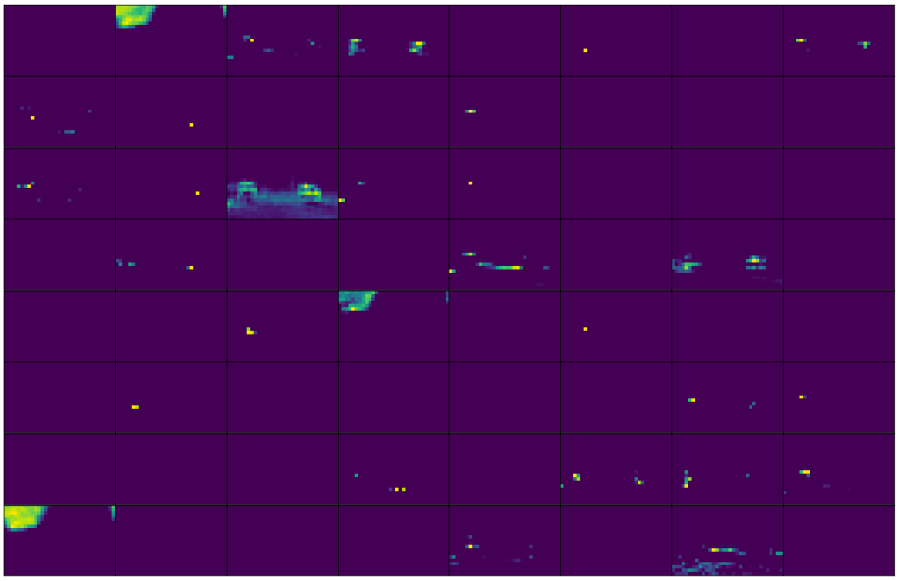
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)

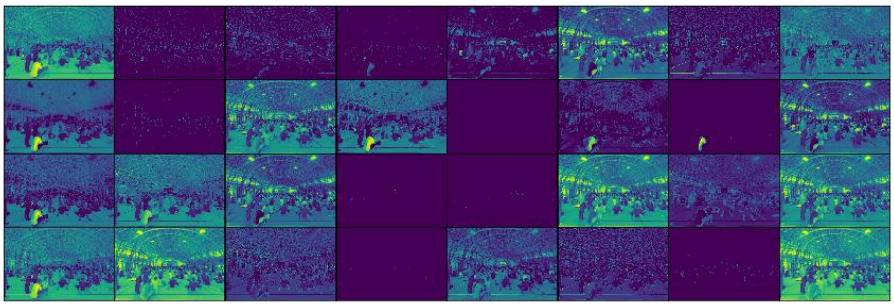


CL10 output: (46, 71, 32)

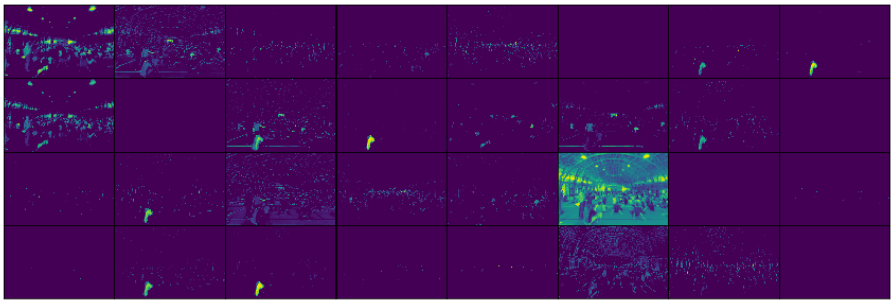


CL13 output: (21, 33, 64)

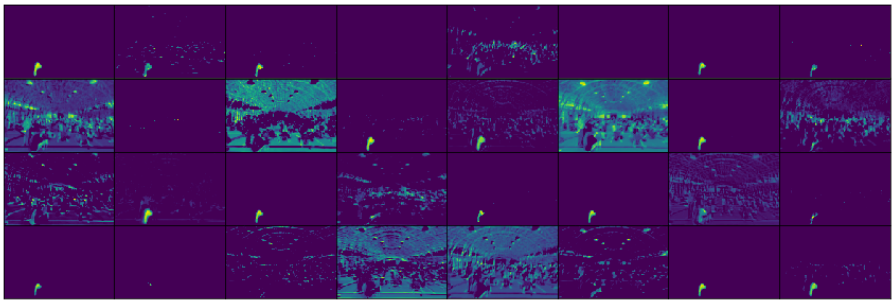
Input image (Fig. 4.1(c)), prediction: 0.039



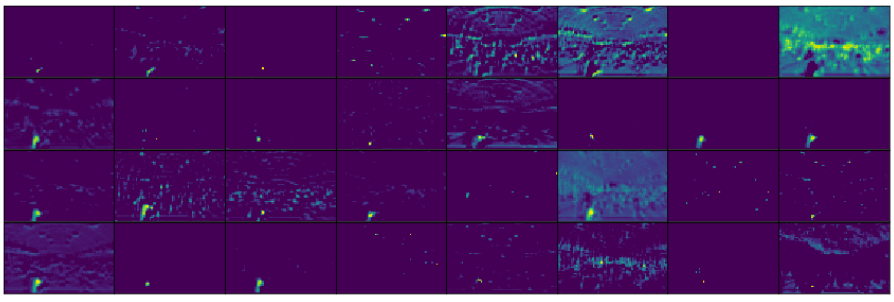
CL1 output: (399, 599, 32)



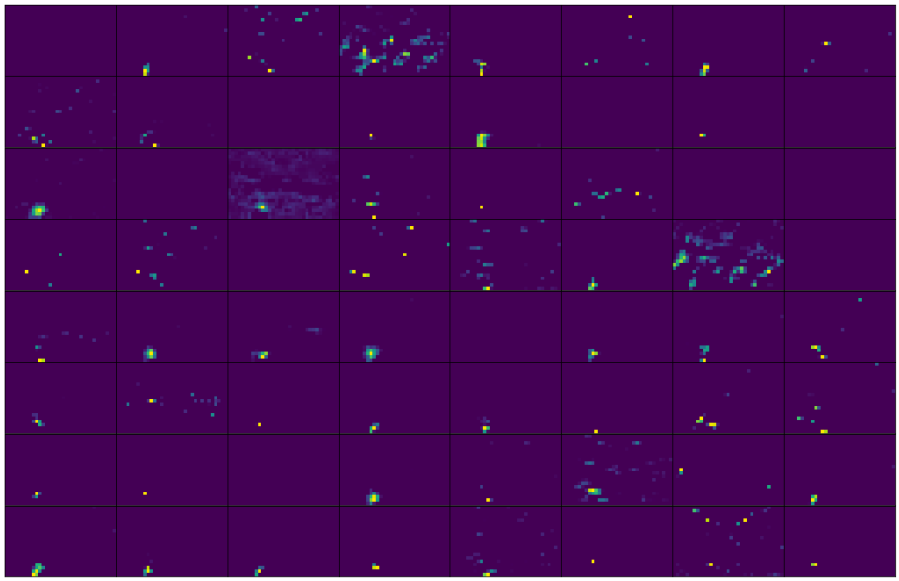
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)

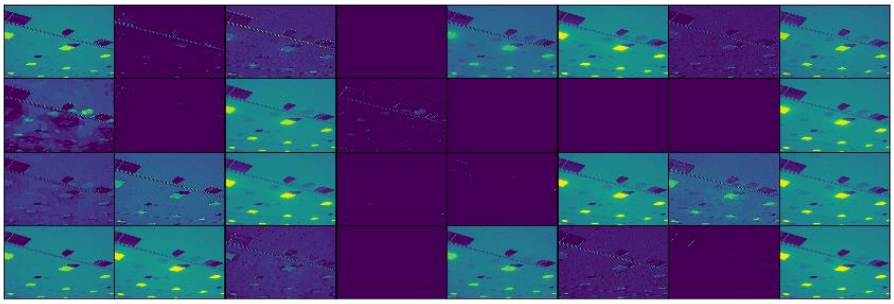


CL10 output: (46, 71, 32)

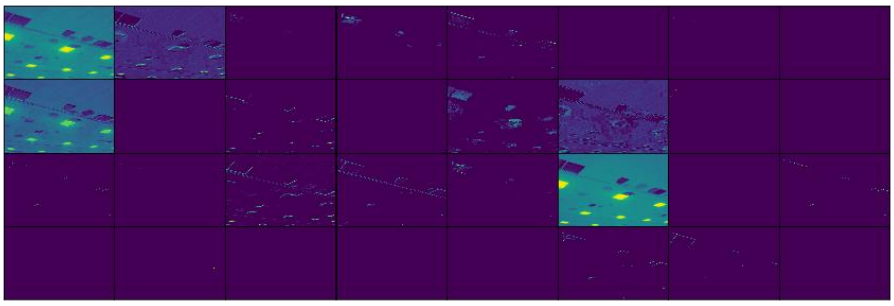


CL13 output: (21, 33, 64)

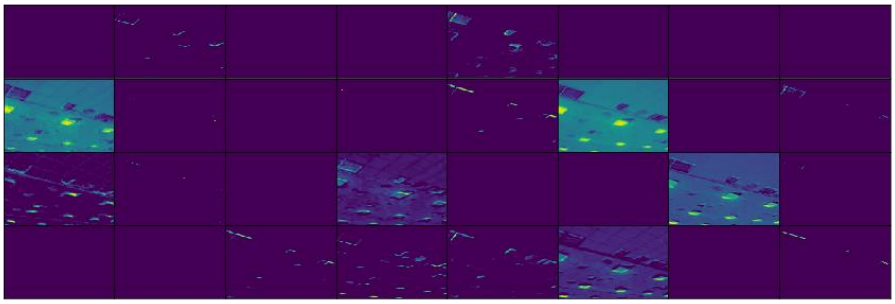
Input image (Fig. 4.1(d)), prediction: 0.737



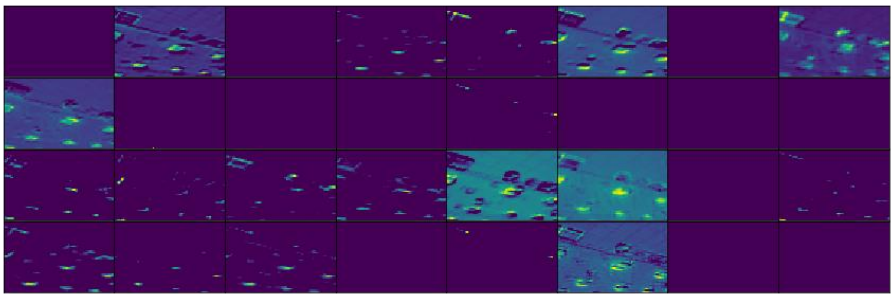
CL1 output: (399, 599, 32)



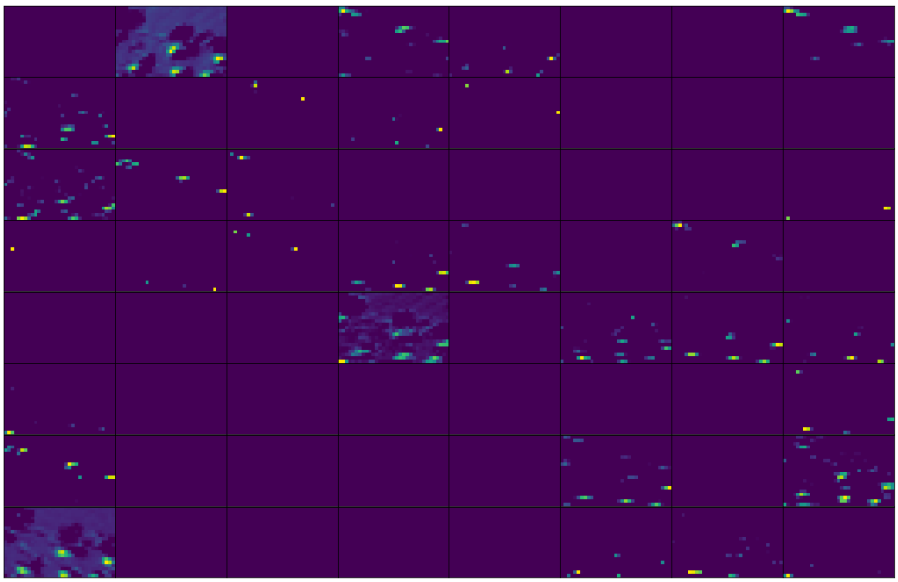
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)

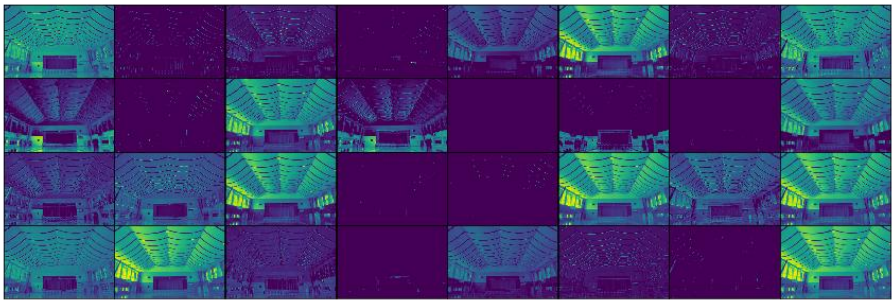


CL10 output: (46, 71, 32)

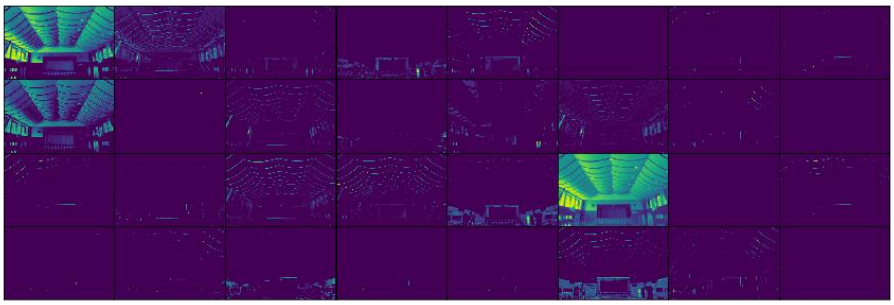


CL13 output: (21, 33, 64)

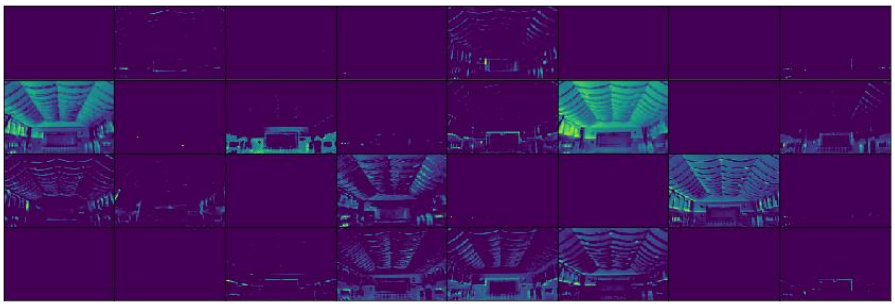
Input image (Fig. 4.1 (e)), prediction: 0.246



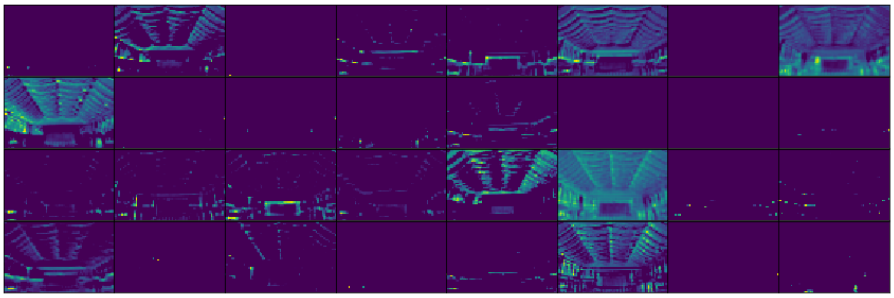
CL1 output: (399, 599, 32)



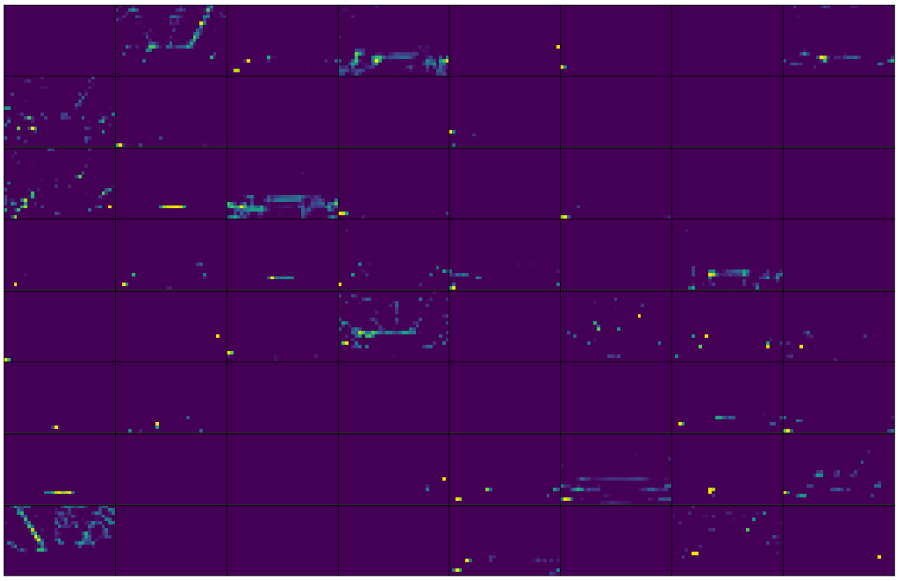
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)

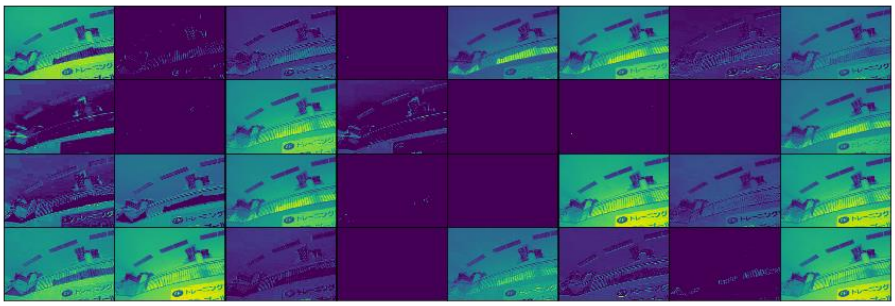


CL10 output: (46, 71, 32)

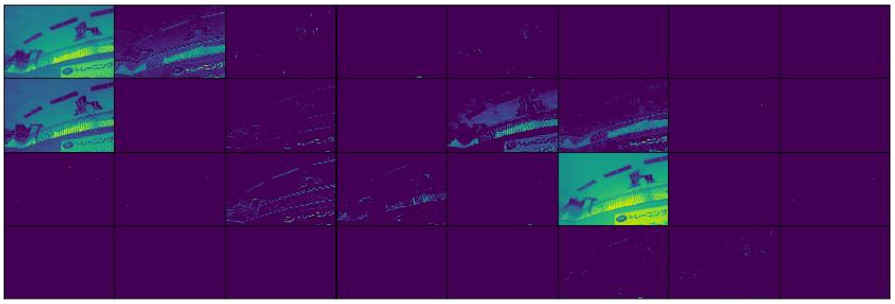


CL13 output: (21, 33, 64)

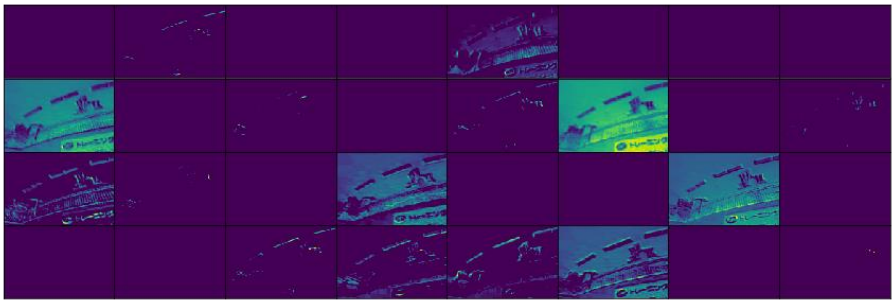
Input image (Fig. 4.1 (f)), prediction: 0.620



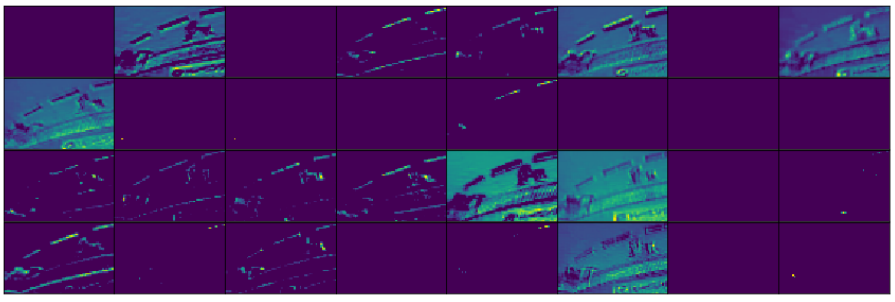
CL1 output: (399, 599, 32)



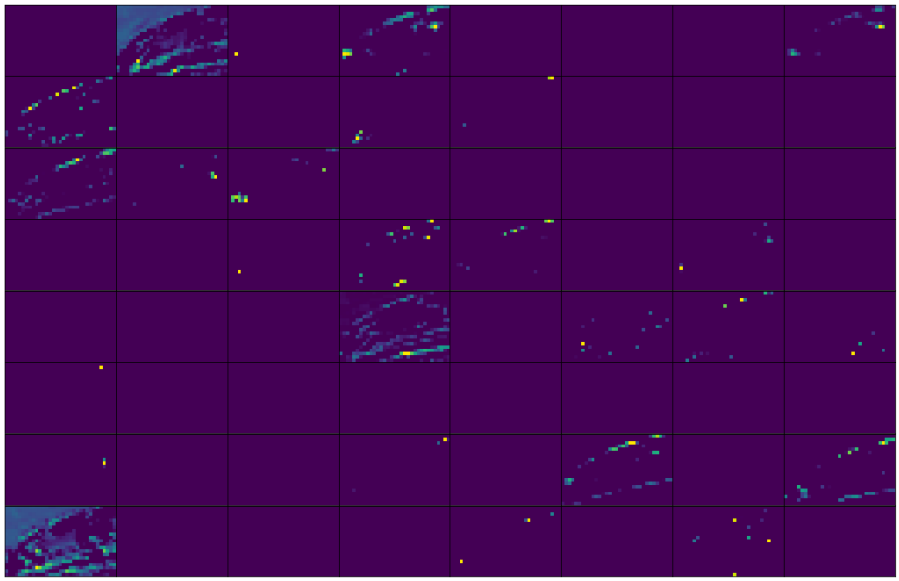
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)

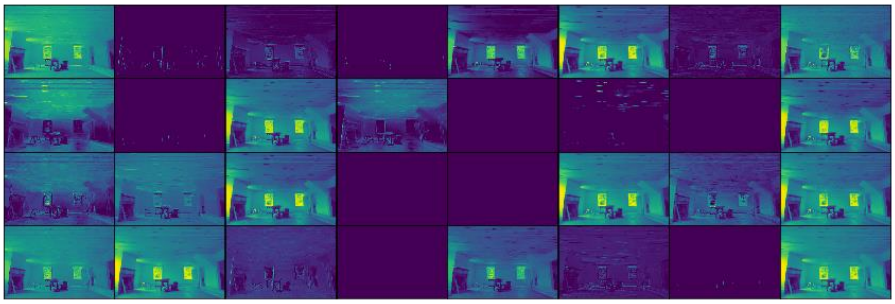


CL10 output: (46, 71, 32)

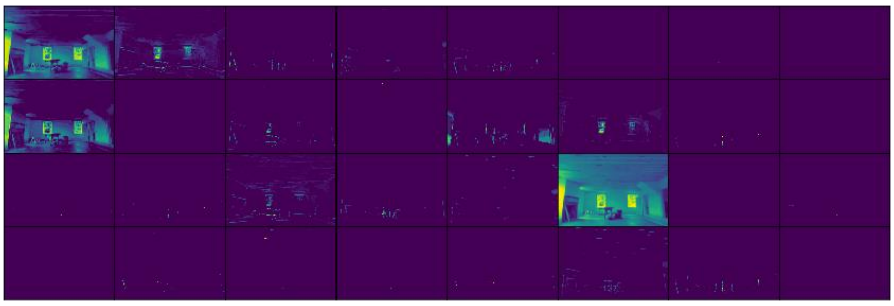


CL13 output: (21, 33, 64)

Input image (Fig. 4.1 (g)), prediction: 0.502

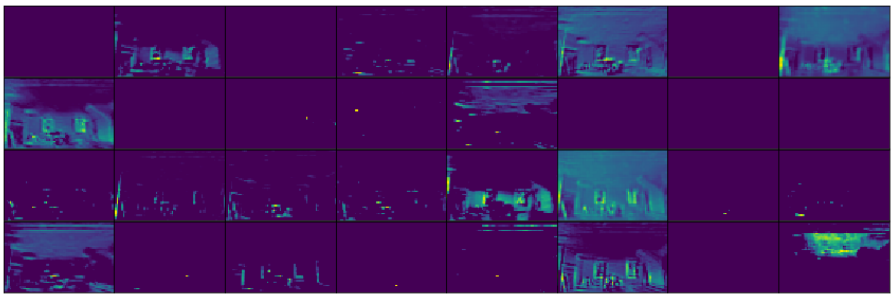


CL1 output: (399, 599, 32)

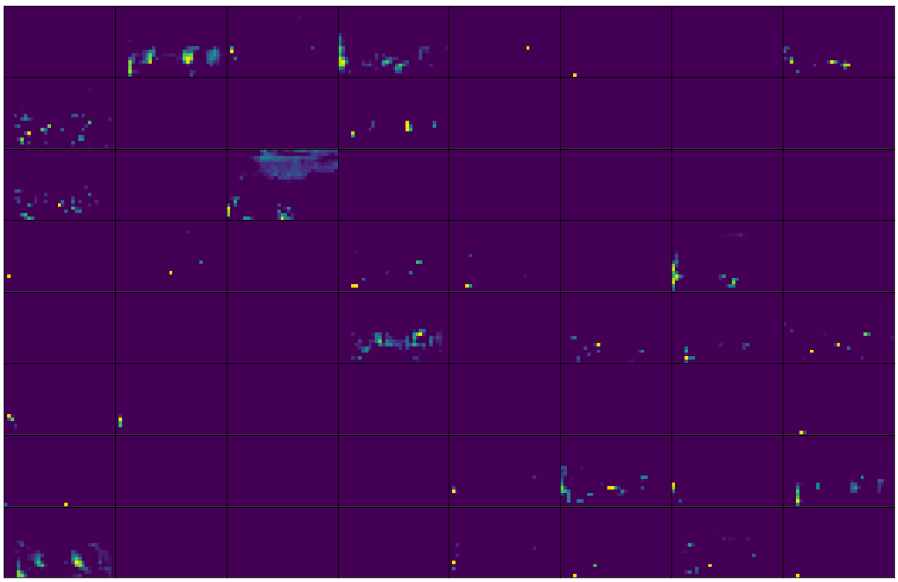


CL4 output: (198, 298, 32)

CL7 output: (97, 147, 32)

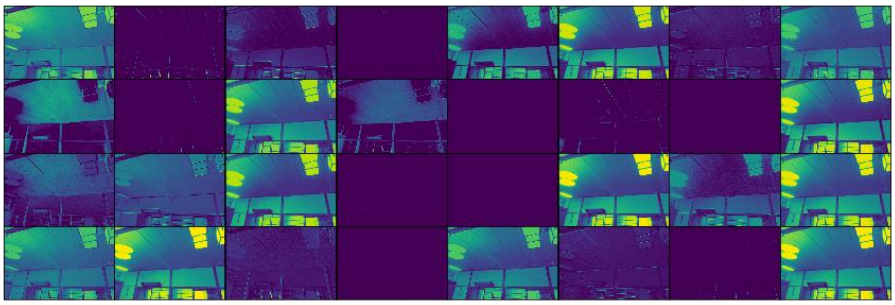


CL10 output: (46, 71, 32)

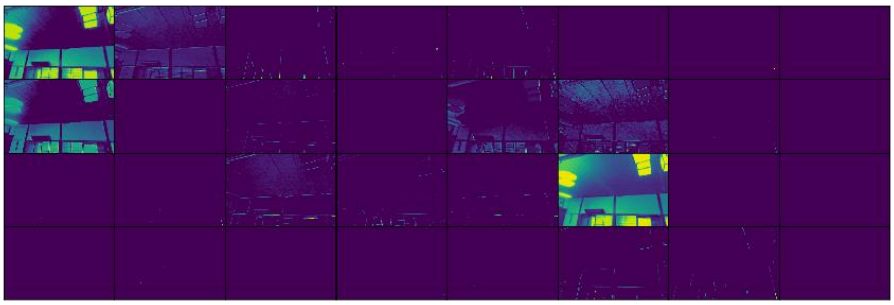


CL13 output: (21, 33, 64)

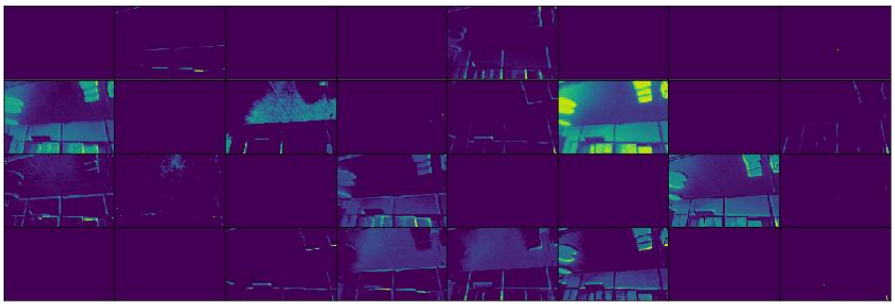
Input image (Fig. 4.1 (h)), prediction: 0.275



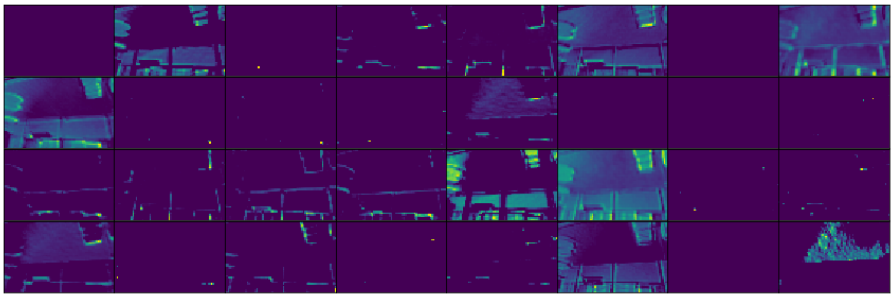
CL1 output: (399, 599, 32)



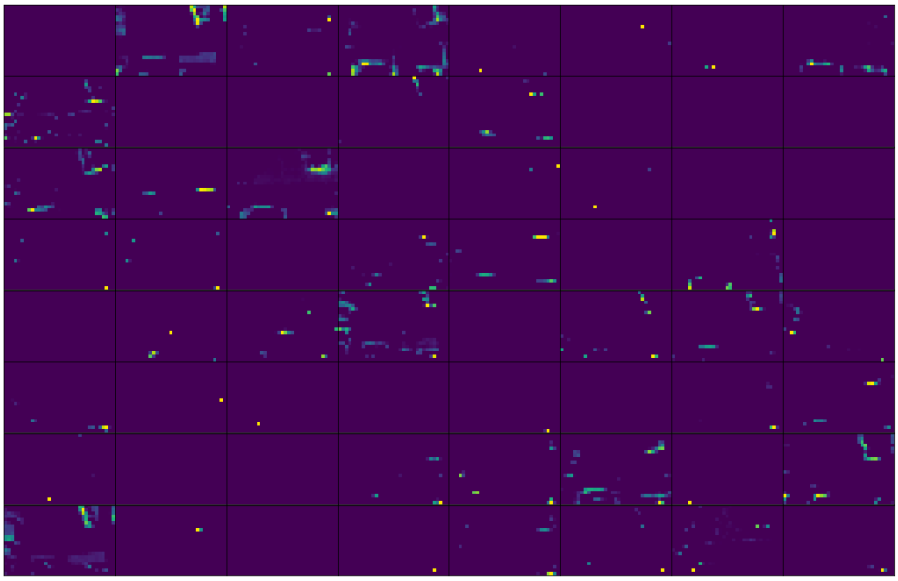
CL4 output: (198, 298, 32)



CL7 output: (97, 147, 32)



CL10 output: (46, 71, 32)



CL13 output: (21, 33, 64)

Acknowledgements

This thesis is the summary of my three-year doctoral research in the University of Tokyo. I would like to give my most sincere thanks to my supervisor, Prof. Ken'ichi Kawaguchi, who has given his wholehearted help, guidance and full trust to me both in my research life and everyday life. His insightfulness and industriousness in research are treasures for my research life. I feel lucky for being supervised by him.

I would like to thank my vice supervisor, A.P. Jun Iyama for his kind help and instructions to my research and the review of this thesis. His comments and guidance to my research summary reports at the end of each semester has provided clear research line. I appreciate the meaningful comments and reviews by Prof. Tsuyoshi Takada, A.P. Tsuyoshi Seike from the University of Tokyo and Prof. Akira Nishitani from Waseda University. Their suggestions help me improve my thesis.

I also would like to give my thanks to the Technical Staff Mr. Shunji Oya who provides hardware support and the secretaries Mrs. Yoko Kondo and Mrs. Mitsuyo Kakimoto in Kawaguchi lab. Research Associate Dr. Yosuke Nakaso has provided me with abundant help and suggestions in both my research and my everyday life. He is always responsive to my inquiry questions and ready to help. Mr. Tatsuya Hiraki, my tutor, has helped me to adapt the daily life in Japan smoothly and rapidly. Mr. Keisuke Mizutani, Mr. Tatsuhiko Hashiba, Mr. Kei Nishizaki, Mr. Tomoki Kawai, Miss. Xitong Yan, Mr. Xuan Yang and et. We have a very good time together.

I give my thanks to Prof. Yingying Zhang, Mr. Tianhao Zhang, Dr. Jianhui Hu and Dr. Lichen Wang, who have spent their time with me during their stay in Kawaguchi Lab. Their comments and advices help me with my research. Miss. Ling Wu and Mr. Hongqi Diao help me a lot with my research and daily life.

I would give my special thanks to my wife, Yiming Xu, who has been supporting and encouraging me for many years. I also thank our parents who have always been supporting us. Without their support and trust I cannot complete my research work.