Crack Detection and Feature Extraction using Deep Learning

A Report Submitted
For B.Tech. Project Part I
By

VISHAL KUMAR (2020CEB021)

Under the supervision of
Prof. Subrata Chakraborty

DEPARTMENT OF CIVIL ENGINEERING

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

December 2023

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to all those who have contributed to the successful completion of my report.

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Prof. **Subrata Chakraborty**, for his invaluable guidance, encouragement, and support throughout the project. I am truly grateful for your patience, encouragement, and constructive feedback. Your mentorship has played a significant role in my academic and personal growth. Your expertise in the field of Civil Engineering and your dedication to my success have been instrumental in shaping my research and have helped me to produce a high-quality report paper.

I would also like to extend my thanks to the faculty members of the Civil Engineering department for their constant support, encouragement, and motivation. Their knowledge, insights, and expertise have been a valuable resource for me during my time at IIEST Shibpur.

Also, I would like to thank Miss Sanchita Das, M. Tech scholar under the guidance of Professor Subrata Chakraborty, who assisted me throughout this project in understanding the concepts.

I would like to express my sincere appreciation to my classmates and friends for their constant support and motivation. Additionally, I would like to express my heartfelt gratitude to my family for their unwavering support, love, and encouragement. Their constant support and encouragement have been a driving force in my academic and personal growth.

Thank you.

VISHAL KUMAR

2020CEB021

7th SEMESTER UG

CONTENT

Page No

CONTENT OF FIGURE

1 INTRODUCTION

Structures, such as bridges, tall buildings, and highways, deteriorate with time, which causes damage to the health of the structures. Structural safety remains challenging in all infrastructure plays a significant role in structural maintenance, and it is exponentially related to its residents. The regular inspection of structures for cracks and damages is important to maintain the safety of people. This project comes under the domain of Structural Health Monitoring (SHM). The SHM has become an important area of research in Civil, Aerospace and Mechanical engineering in recent times concerning development and growth of Machine Learning and deep Learning technologies.

There are various parameters that ensure structural health, cracks remain as key parameters directly indicating the condition of structures. So detection of cracks through the technologies with good accuracy may ensure the safety of residents and prevent accidents. Thus the difficulties faced in implementing traditional procedures and the need to develop computer-based automated evaluation process, has motivated the application of various types of soft computing tools in recent times. With the advancement in digital technology, in Civil Engineering and other departments, the use of DL and computer vision tasks like Edge Detection and threshold have become very popular in recent times.

Keeping a structure healthy using computer vision and machine learning is now an autonomous process with just digital images. This also reduces human error and dependency in the SHM field. There are some Non-Destructive methods employed for crack detection like ultrasonic testing, infrared and thermal testing, radiographic testing, laser testing and image processing. Considering all of these approaches, image processing techniques stand out as being quite accurate and low-cost. The main objective of crack detection is to control crack parameters such as length, width, and cracks pattern. A Neural Network model is created for the classification of cracked & uncracked surface images. Then some methods are further applied to find crack parameters.

2 Creating a Model

For making models the most difficult task is collecting the dataset. There are some researchers who contribute large datasets in different platforms, such as kaggle. This is very important to work with large datasets while using Neural Network technologies to get good accuracy at time to training and also testing. Further, collecting the dataset which is labelled in required category is cracked surface and uncracked surface. This is an image based binary classification problem.

Convolutional Neural Network is the most used NN for image classification problems, object detection and so on. Nowadays CNN is used in various fields in Civil Engineering or other departments like in structural, transportation and environmental engineering. There are also very powerful and famous pre-trained CNN that are ResNet, VGG-19, Inception-V3 and so many.

There are some important skills which are relevant steps for making proper CNN model:

- Gathering dataset
- Setup Tensorflow and Keras framework platform (google Colab)
- Import Python and their libraries like OpenCV, matplotlib etc.
- Image processing includes splitting of datasets in the train and test folder.
- Study of CNN Architecture
- Appling Edge Detection concepts (Crack segmentation)
- Import Pre-trained model that is ResNET
- Understand and apply Transfer Learning concepts for crack datasets
    - Feature Extraction
    - Fine Tuning
- Train and test the Model
- Play with a new crack image
- Visualise accuracy

Each size of image in the dataset is necessary to be the same size or same pixel before training the model using this image. The total number of images is about 40,000 which is sophisticated to train the model and get good accuracy. The dimension of each image is 227 x 227 in the collected dataset. And this 227 x 227 size image is called input size image which is mentioned in the CNN code.

2.1 Datasets

The size of the training dataset needs to be adequately large to achieve optimum results without the issue of overfitting the data (means memorization of data by machine instead of generalising the inherent features of the data). This dataset is taken from the website Mendeley Data - Crack Detection, contributed by “Çağlar Fırat Özgenel”.

The datasets contains images of various concrete surfaces with and without crack. The image data are divided into two as negative (without crack) and positive (with crack) in separate folder for image classification. Each class has 20000images with a total of 40000 images with 227 x 227 pixels with RGB channels. The dataset is generated from 458 high-resolution images (4032x3024 pixel) with the method proposed by *Zhang et al (2016)*. High resolution image found out to have high variance in terms of surface finish and illumination condition. No data augmentation in terms of random rotation or flipping or tilting is applied.

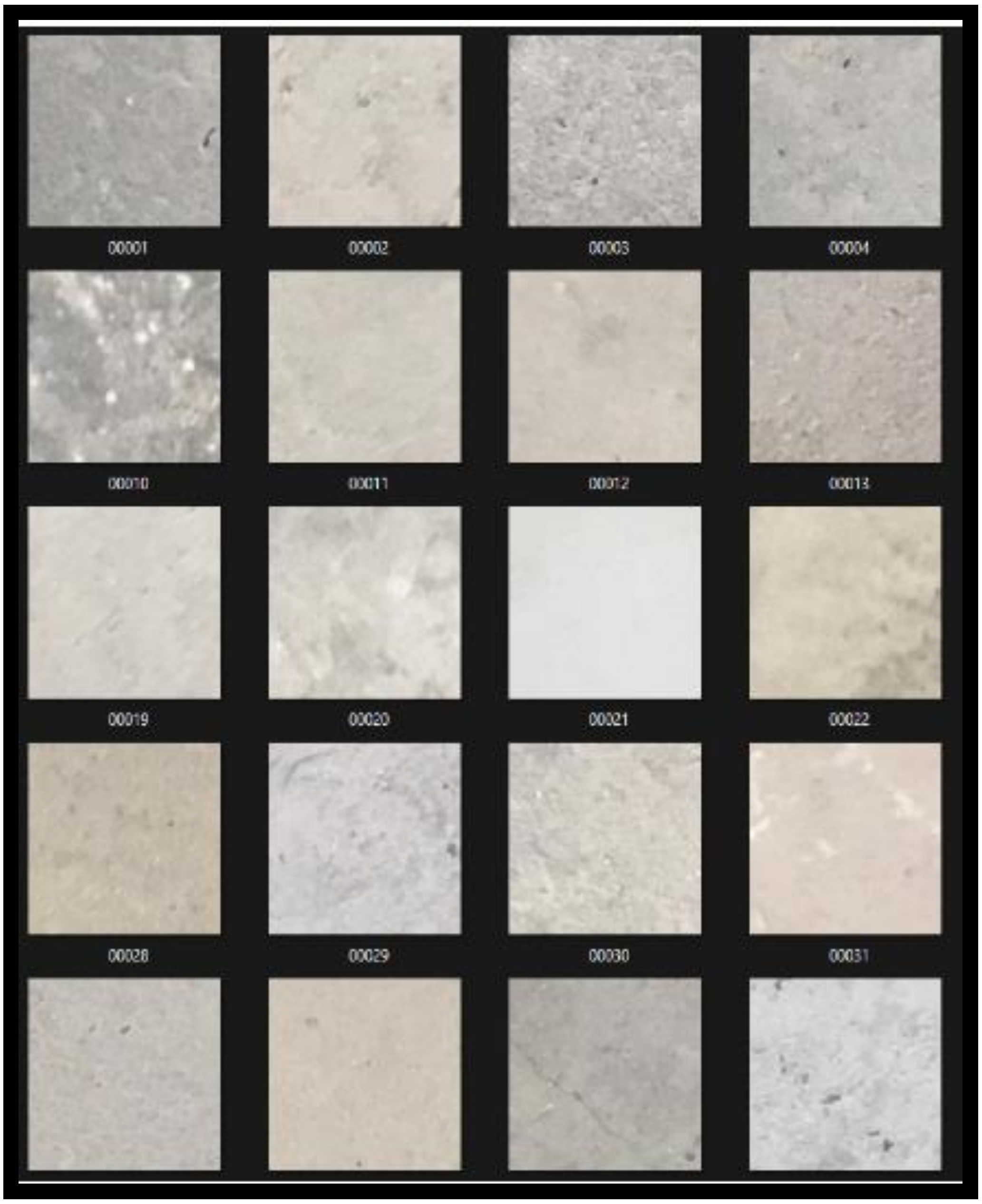


Fig. 2.1 Uncracked surface Images (Negative)

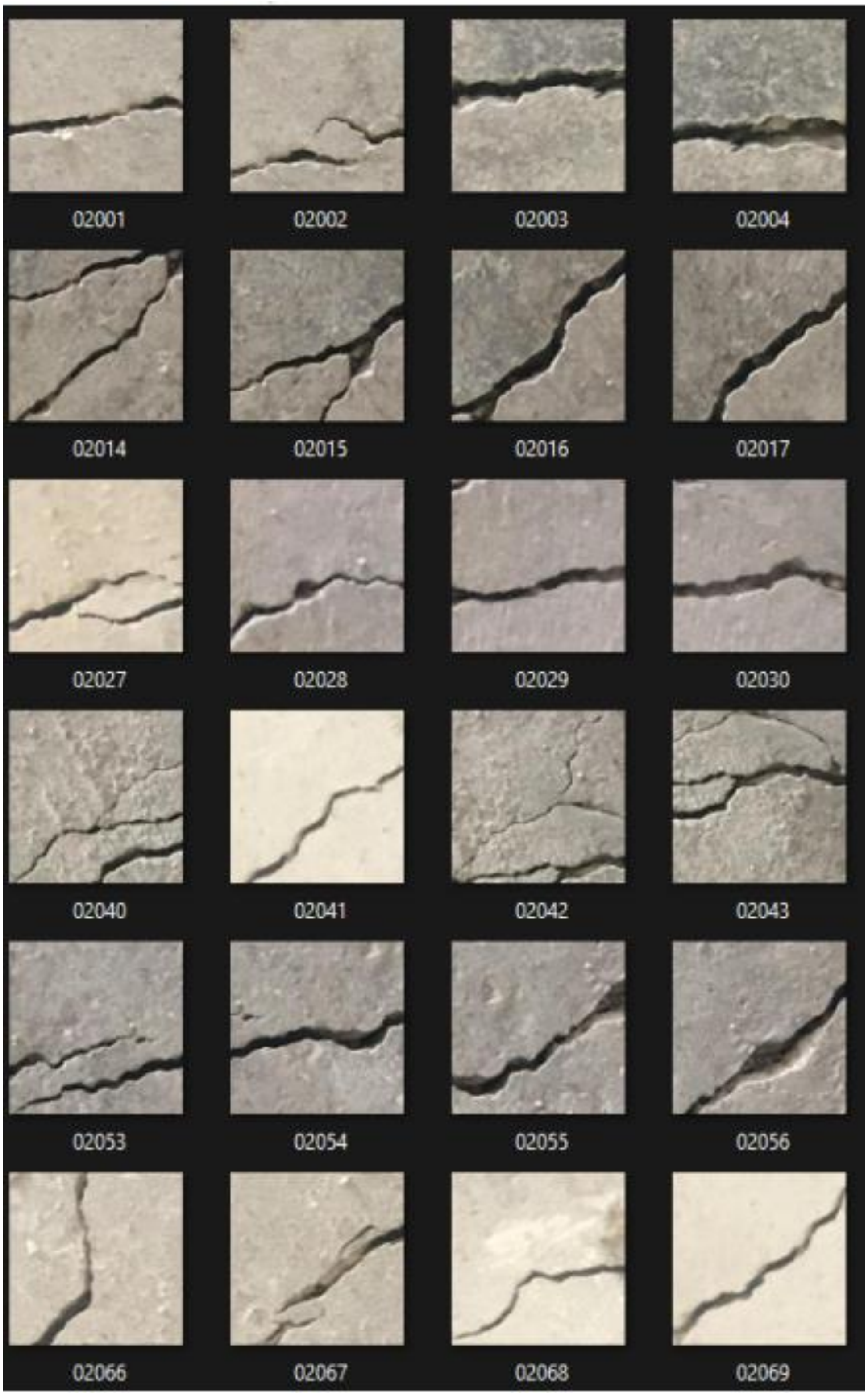


Fig. 2.2 Cracked Surface Images (Positive)

2.2 Neural Network (NN)

Before talking about CNN first understand Neural Network. A neural network is a computational model inspired by the way biological neural networks in the human brain work. It is composed of interconnected nodes, also known as neurons or artificial neurons, organized into layers. Neural networks are a fundamental component of machine learning and artificial intelligence, and they can be used for a wide range of tasks, including pattern recognition, classification, regression, and more. A simple neural network have basic three layers that is Input layer, multiple hidden layer and the final layer is Output layer.

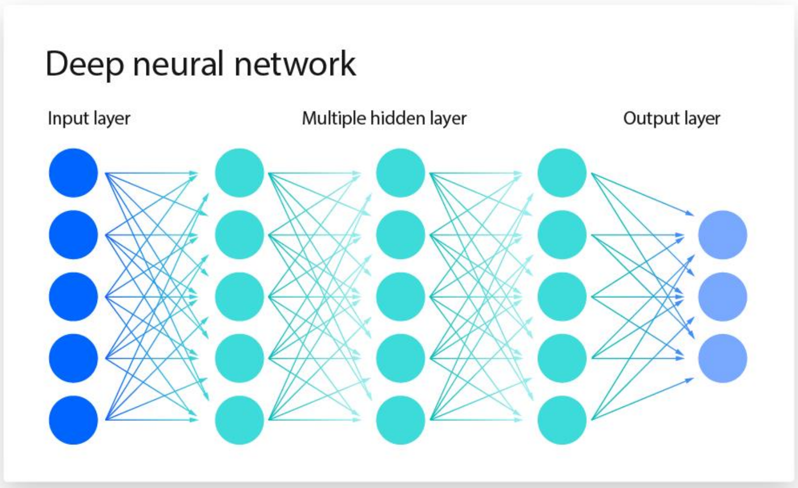


Fig. 2.3 Deep Neural Network

There are some important parameters and concepts of Neural Network that is neuron , weights and biases, gradient, activation function, backpropagation, loss function, normalization, overfitting and underfitting.

A. Weights and Biases

Weights are the real values that are attached with each input/features and they convey the importance of that corresponding feature in predicting the input output. Each connection between two neurons has an associated weight(w). During the training process, these weights are iteratively updated to minimize the differences between the predicting output and the actual target.

The learning algorithm (often gradient descent or a variant) adjust these weights based on the error and accuracy between the predicted output and the actual target, thus allowing the network to learn from the training data.

Biases are additional parameters in each neuron that are independent of the input. They provide the network with the flexibility to shift the output in a way that is not solely dependent on the input value. Every neuron in a layer typically has its own bias. These are important for ensuring that the network can learn and represent complex relationships, even when the input values are not sufficient to produced desired output without an additional shift. Similar to weight, biases are learned during training. The learning algorithm adjust them based on the error between the predicted output and the actual target.

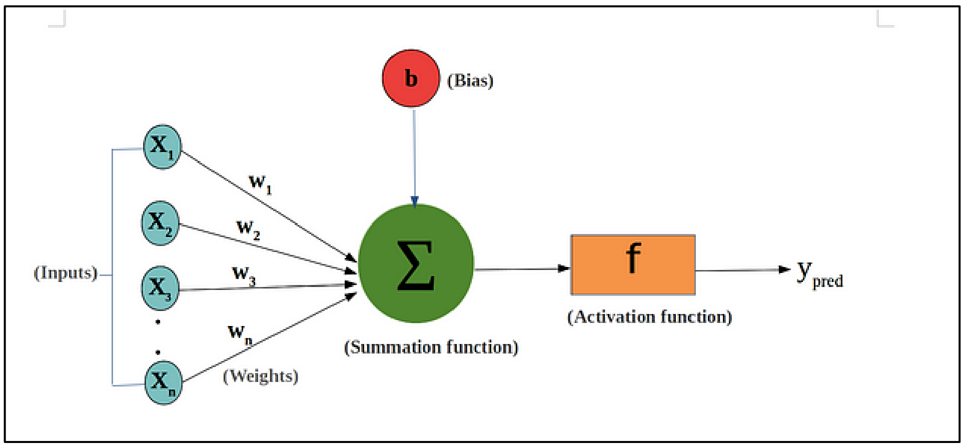


Fig. 2.4 Basic Propagation of Neural Network

Mathematically, the computation in a neuron can be represented as follows:

$$\text{Output} = \text{Activation} \left(\sum (\text{input} \times \text{weight}) + \text{bias}\right)$$

Here,

- Activation is the activation function applied to the weighted sum of inputs and biases.
- $\sum$ denotes the sum over all input values.
- input represents the input values.
- weight represents the corresponding weights.
- bias represents the bias term.

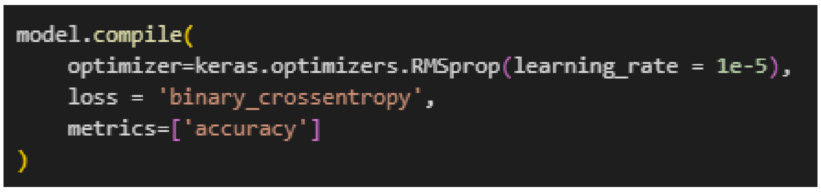
B. Loss function

The loss function is a measure of how well the model predicts the true labels for each training example. In this project using the most commonly used loss function is the binary cross-entropy loss function (also known as log loss). The formula for this function is:

$$L(y, \hat{y}) = -[y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})]$$

Where:

- L is the loss value
- y is the true label (0 or 1) for the training example
- ŷ is the predicted probability of the positive class (usually obtained using the sigmoid function) for the training example



```
model.compile(
    optimizer=keras.optimizers.RMSprop(learning_rate = 1e-5),
    loss = 'binary_crossentropy',
    metrics=['accuracy']
)
```

Fig. 2.5 binary_crossentropy

C. Cost function

The cost function, also known as the objective function or the average loss, is the average value of the loss function over the entire training dataset. It represents the overall performance of the model across all training examples. The purpose of the cost function is to guide the learning process and find the optimal set of model parameters (weights(w) and biases(b)) that minimize the prediction errors.

$$J(w, b) = (1/m) * \sum[L(y, \hat{y})]$$

Where:

- J is the cost function
- w is the vector of model weights
- b is the bias term
- m is the number of training examples
- $L(y, \hat{y})$ is the loss function for each training example

D. Gradient Descent

Gradient descent is a mathematical optimization algorithm used to find the minimum value of a function. It works by iteratively adjusting the parameters of the function in the opposite direction of the gradient, which points in the direction of steepest ascent. This process is repeated until a minimum value is reached or a specified number of iterations have been completed. Gradient descent is widely used in machine learning and AI applications to optimize model parameters and improve performace of the ML and DL based model.

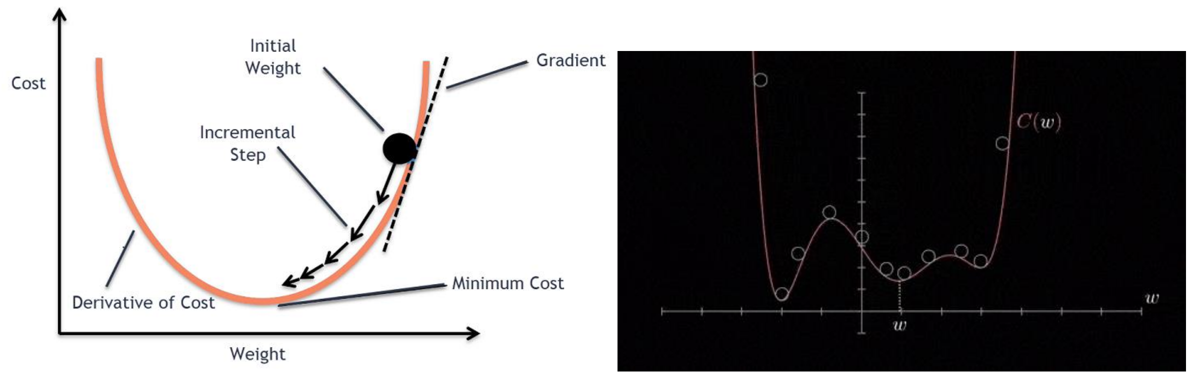


Fig. 2.6 Gradient descent example

E. Activation Function

An activation function is a mathematical operation applied to the output of a neuron in a neural network. It introduces non-linearity to the network, which allow NN too compute complex functions. It plays a vital role in the training process of NN. During backpropagation, which is process of updating the network's weights.

There are some common activation functions used in neural networks :

1. Sigmoid Function :
2. ReLu Function :
3. tanh Function :

4. Sigmoid Function :

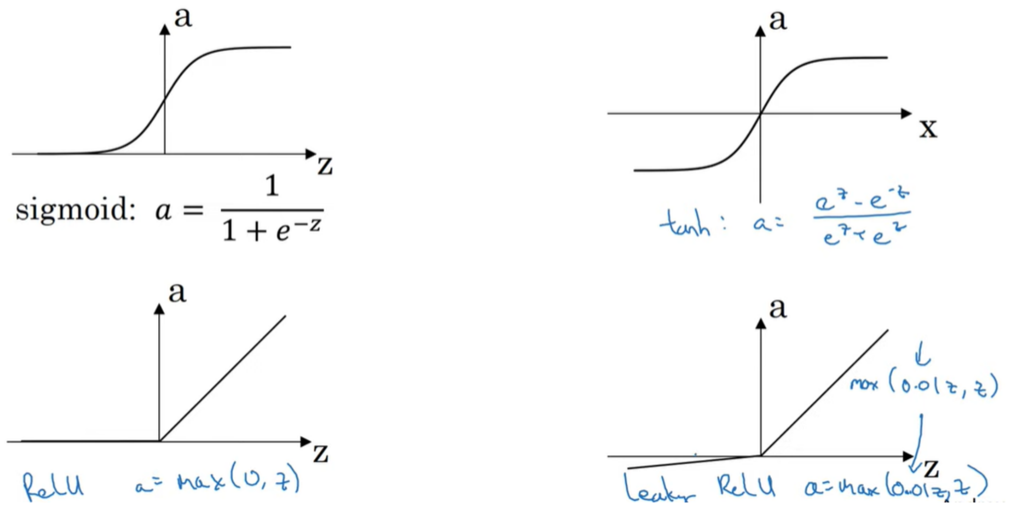

a
z
sigmoid: $a = \frac{1}{1+e^{-z}}$
a
x
tanh : $a = \frac{e^z - e^{-z}}{e^z + e^z}$
a
z
ReLU $a = max(0, z)$
a
max (0.01z, z)
z
Leaky ReLU $a = max(0.01z, z)$

Fig. 2.7 Activation Function

F. Overfitting Problems and solution

Overfitting is a common problem in machine learning where a model is trained too well on the training data and performs poorly on new data. It occurs when a model is too complex and has learned the noise in the training data instead of the underlying pattern. This means that the model will perform well on the training data but poorly on new data.

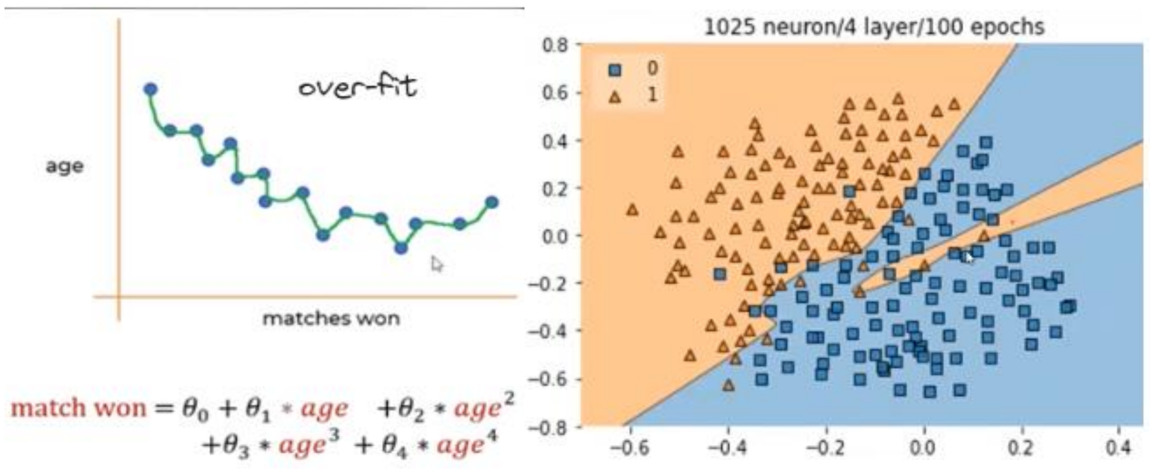


Fig. 2.8 Overfitting problem diagram

This shows model try to connect all the datasets and make graph complex. So very oftenly occuring issue which is called overfitting. To avoid Overfitting, it is important to use techniques such as regularization and cross-validation. Regularization involves adding a penalty term to the loss function to prevent overfitting and to reduce the high bias. There are two common type regularization:

L1 regularization add penalty term proportional to the absolute value of the weights. L2 regularization add penalty term proportional to the square of the weights.

G. Dropout method

It involves randomly droping out some nodes during training to prevent the network from reyling too heavily on any one node. In figure below, the neural network on left represent a typical neural network where all units are activated. On the right, the red units have been dropped out of the model – the values of their weights and biases are not considerd during training. These helps in avoid the problem of overfitting.

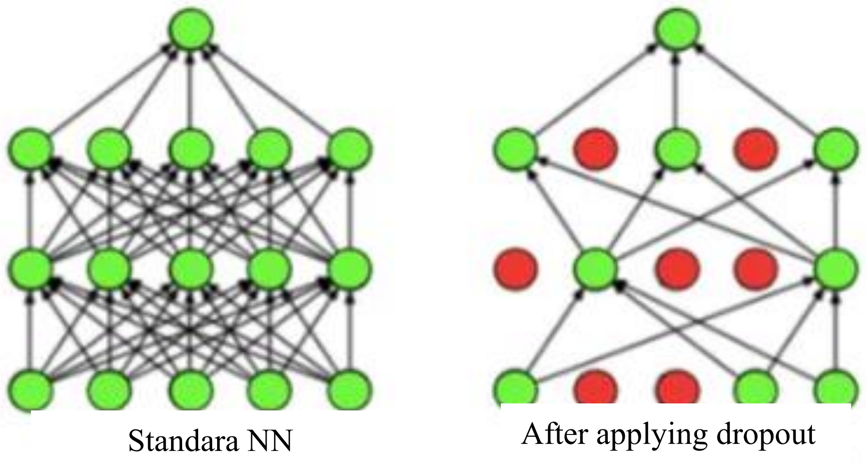


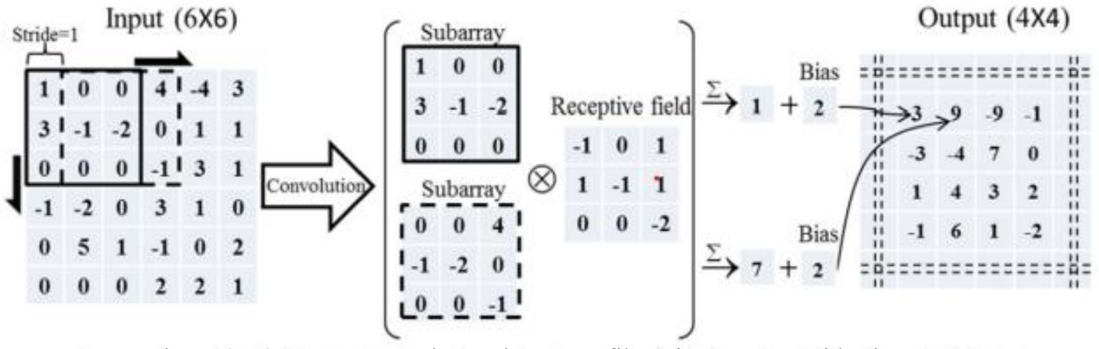
Fig. 2.9 Dropout Example

There some famous other Regularization Methods :

1. Data Augmentation
2. Early stopping
3. Normalization Inputs
4. Vanishing and Exploding

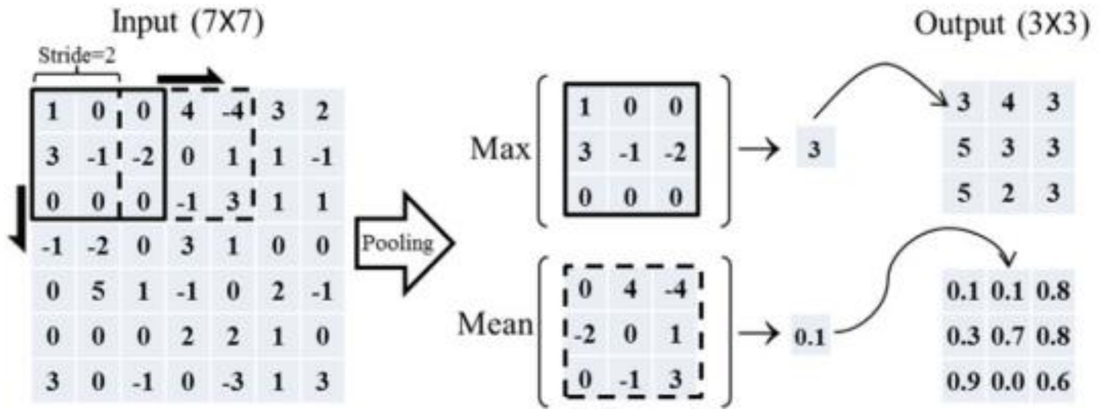
2.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs have played a crucial role in the advancement of computer vision and have become a cornerstone in many state-of-the-art machine learning models for visual tasks. It is a class of deep neural networks (DNN) that has proven to be very effective in areas such as image classification, object detection, pose detection, localizations and video recognition and computer vision tasks. A convolution is an operation which is done between the input array and applied different filter layers. First, it performs element-by-element multiplications (i.e. dot product) between a subarray of an input array and a receptive field. These receptive filed is also often called a filter, or kernel. The initial weight values of a kernel are typically randomly generated. The size of a subarray is always equal to a filter field, but a receptive field is always smaller than the input layer. Second is added to summed values. Fig shows the convolution of the subarrays



Output size = (I-R) / S + 1, I = input size, R = filter size, S = stride size ;

Fig. 2.10 Convolution Example



Output size = (I-P) / S + 1, I = input size, P = Pooling size, S = stride size ;

Fig. 2.11 Pooling Example

CNNs use a specialized architecture to automatically and adaptively learn spatial hierarchies of features from input data. A CNNs model built in presence of some important layers that is convolutional layer, activation layer, pooling layer, SoftMax layer and so on. In Fig. 12. Mention the overall architecture of CNN with all layers.

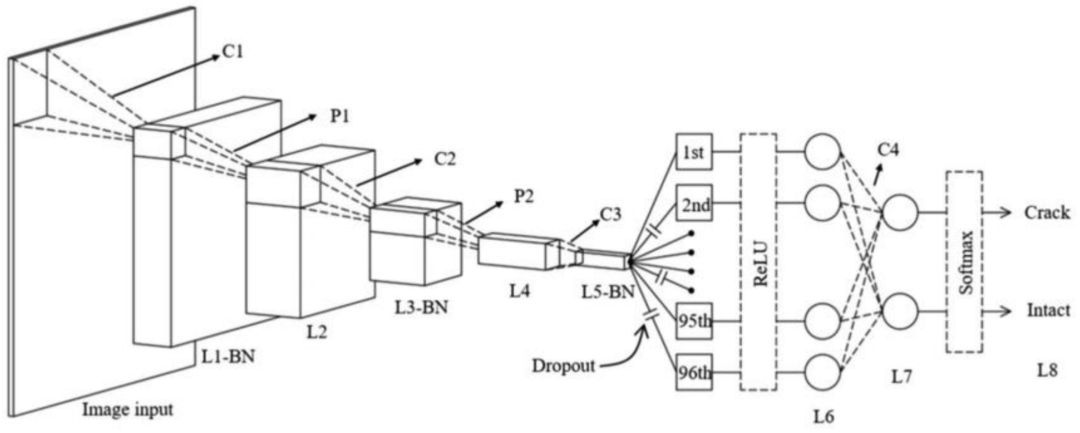
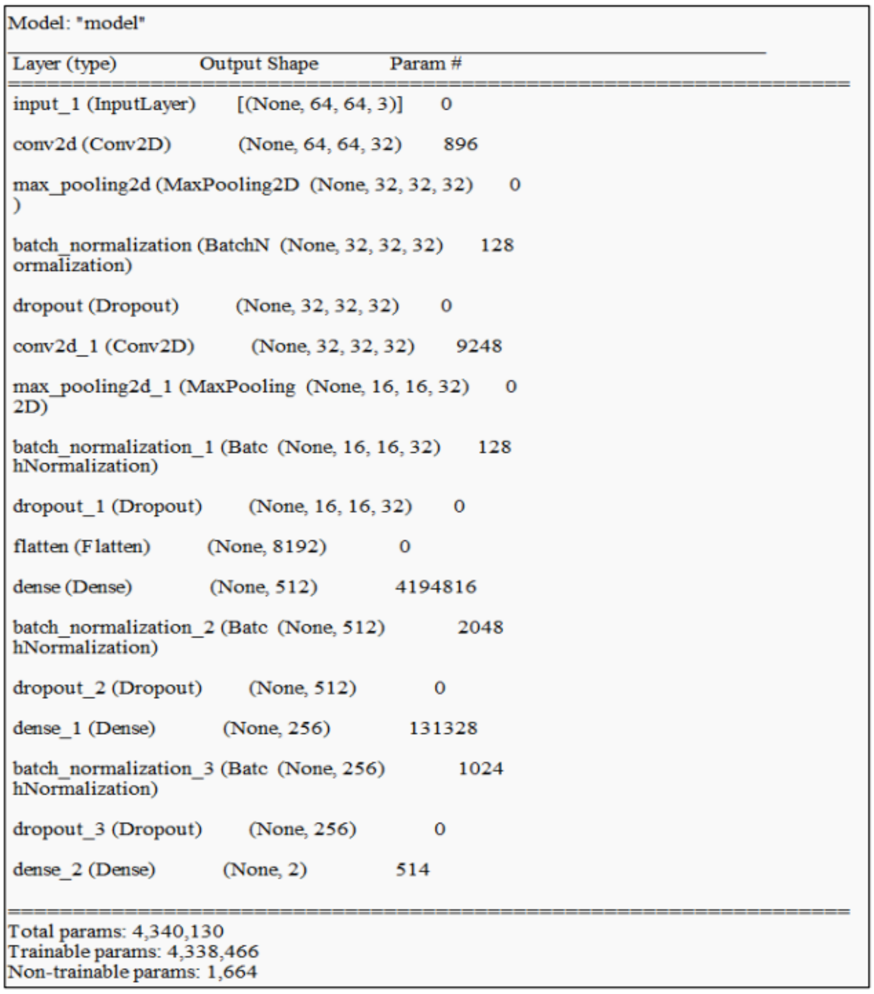


Fig. 2.12 Overall architecture

L#: Layes corresponding to operation (L1, L3, L5 and L7: conv2d layers; L2 and L4: max pooling layers; L6: ReLu activation function layer; L8: softmax layer); C# : convolution: P# : pooling ; BN ; batch normalization.

In Figure 13, show the sample summary of crack identification model summary for Convolutional Neural network. And this sample model summary for Input image 227 x 227.



```
Model: "model"
_________________________________________________________________
 Layer (type)              Output Shape           Param #
=================================================================
 input_1 (InputLayer)      [(None, 64, 64, 3)]    0

 conv2d (Conv2D)           (None, 64, 64, 32)     896

 max_pooling2d (MaxPooling2D  (None, 32, 32, 32)    0
 )

 batch_normalization (BatchN  (None, 32, 32, 32)    128
 ormalization)

 dropout (Dropout)         (None, 32, 32, 32)     0

 conv2d_1 (Conv2D)         (None, 32, 32, 32)     9248

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)    0
 2D)

 batch_normalization_1 (Batc  (None, 16, 16, 32)    128
 hNormalization)

 dropout_1 (Dropout)       (None, 16, 16, 32)     0

 flatten (Flatten)         (None, 8192)           0

 dense (Dense)             (None, 512)            4194816

 batch_normalization_2 (Batc  (None, 512)           2048
 hNormalization)

 dropout_2 (Dropout)       (None, 512)            0

 dense_1 (Dense)           (None, 256)            131328

 batch_normalization_3 (Batc  (None, 256)           1024
 hNormalization)

 dropout_3 (Dropout)       (None, 256)            0

 dense_2 (Dense)           (None, 2)              514

=================================================================
Total params: 4,340,130
Trainable params: 4,338,466
Non-trainable params: 1,664
```

Fig. 2.13 Summary of CNN model for crack identification

2.4 PRE-TRAINED CNN MODEL

This pre-training is typically done on massive dataset, like ImageNet, which contains millions of labelled images across thousands of categories. The idea behind pre-training is that the features learned by the CNN on the large dataset are generic and can be possible transfer learning on a smaller dataset specific to your own project, saving a considerable amount of time and computational resources compared to training a model from scratch. And also the chance of getting a good accuracy is enhanced. There are some popular deep learning frame works which give empowerment to use this pre-trained NN that is Tensorflow, keras and Pytorch. This approach has become a standard practice in the development of state-of-the-art models for various computer vision applications.

There are some famous pretrained CNN model which are top competitors in ILSVRC. The ImageNet Project is a large visual database designed for use in visual object recognition software research. The ImageNet project runs an annual software contest, the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC**), where software programs compete to correctly classify and detect objects and scenes. Below Fig 14 represent the statistic of some top winner pre-trained neural network with their number of layers and error percentage.

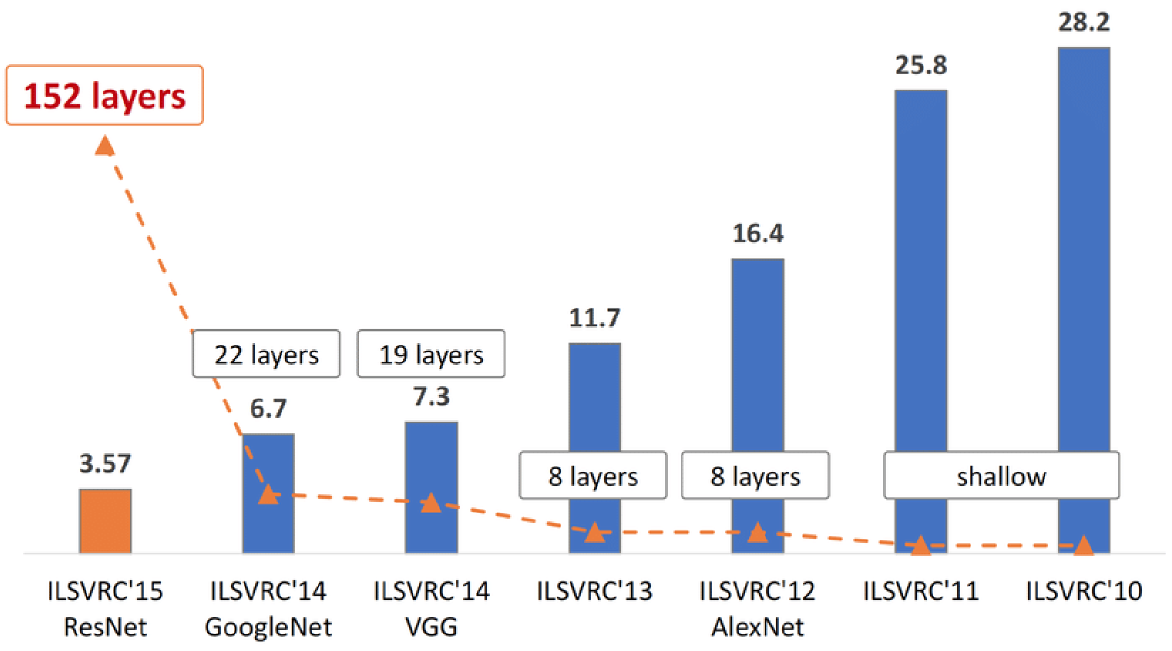


Fig. 2.14 The evolution of the winning entries on the ILSVRC 2010 to 2015

Since 2012, CNNs have out performed hand-crafted descriptions and shallow networks by a large margin. The paper by Nguyen, Fookes, Member, Ross, Sridharan (2017). The most widely used pre-trained model in research work is ResNet, VGG-19 and sometimes Inception.

2.5 ResNet (Residual Neural Network)

At last, at the ILSVRC 2015 ,The Residual Neural Network (ResNet) is introduced a novel architecture with “skip connections”. The paper by He, Zhang, Ren and Sun (2015). It achieves a top-5 error rate of 3.57% which beats huma -level performance on this dataset. ResNet-50, ResNet-101 and so many are the popular architectures. It can scale up to hundreds of layers with degradation in performance.

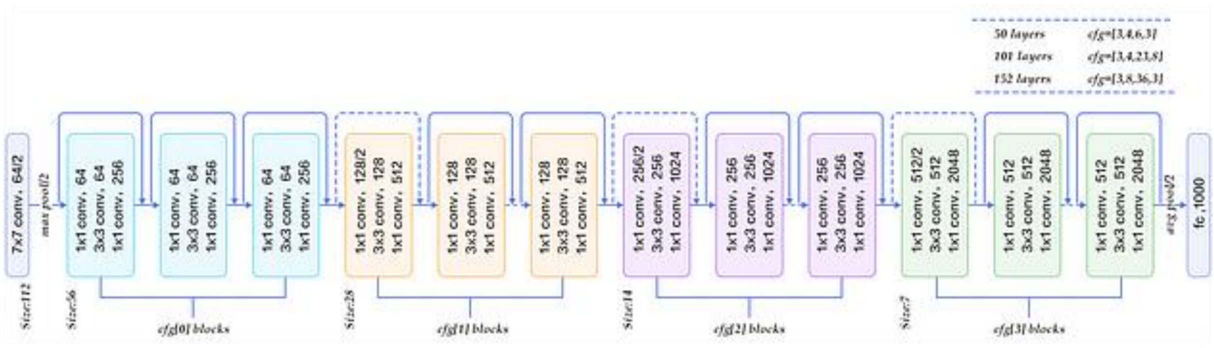


Fig. 2.15 Network structure of ResNet18

Fig. 15. is the network structure of ResNet18. There are 8 basic block modules in the red dotted area, and each module's blue arrow is a short connection. This means that the neural network can be deeper because the gradient can transfer farther during backpropagation. The model is divided into 20 layers, including 17 convolutional layers and three fully connected layers. The pooling layer is not shown in the figure but exists after each convolutional layer. The specific parameters of the ResNet18 model are the same as those of the VGGNet13 model.

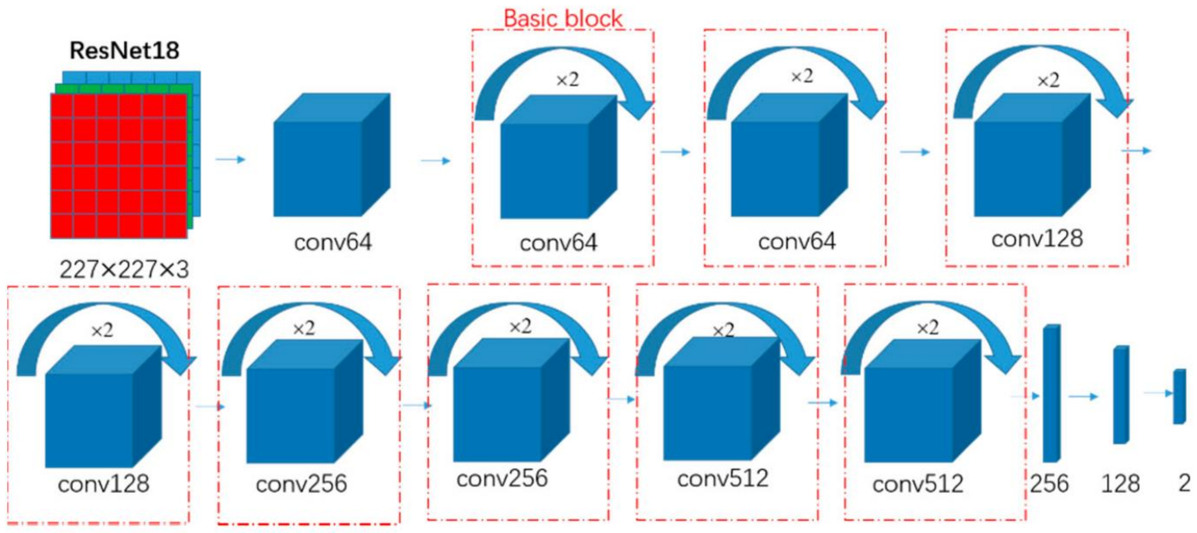


Fig. 2.16 ResNet Block Architecture

2.6 VGG-16 Neural Network

VGG16 is a convolutional neural network trained on a subset of the ImageNet dataset, a collection of over 14 million images belonging to 22,000 categories. K. Simonyan and A. Zisserman proposed this model in the 2015 paper, Very Deep Convolutional Networks for Large-Scale Image Recognition. In the 2014 ImageNet Classification Challenge, VGG16 achieved a 92.7% classification accuracy.

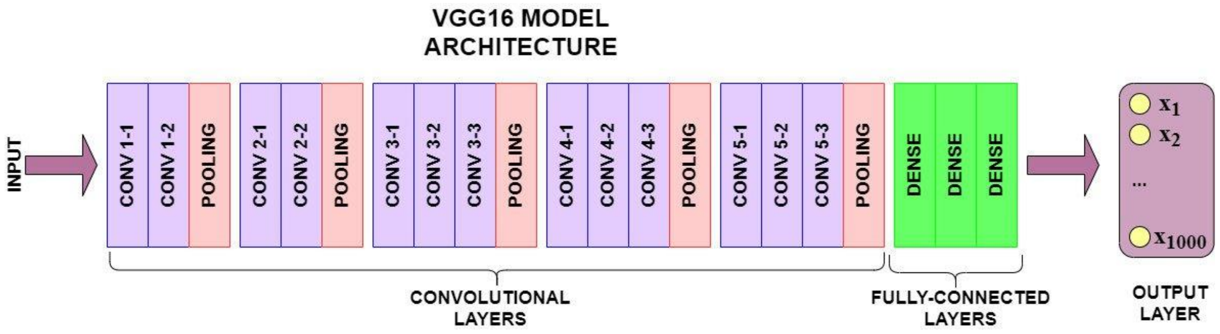


Fig. 2.17 VGG16 model

The model in Fig. 2.17 has 16 convolutional and max pooling layers, 3 dense layers for the fully-connected layer, and an output layer of 1,000 nodes. But more importantly, it has been trained on millions of images. Its pre-trained architecture can detect generic visual features present in our Food dataset. There are many images of two kinds of cars: Ferrari sports cars and Audi passenger cars. A model that can classify an image as one of the two classes. Writing our own CNN is not an option since do not have a dataset sufficient in size. Here's where Transfer Learning comes to the rescue. The ImageNet dataset contains images of different vehicles. It can import a model that has been pre-trained on the ImagNet dataset and use its pre-trained layers for feature extraction.

2.7 Transfer Learning

Transfer learning is an approach in which we use generic pretrained model and retain it on our own dataset. It involves taking a pre-trained neural network (like resNet, vgg or a model that already proven or trained successfully on large dataset for a specific task) and using it as the starting point for a new task. The basic idea behind transfer learning is that the knowledge gained from the first task can be beneficial for second task. Multiple deep learning domains use this approach, including Image Classification, Natural Language Processing, and even Gaming! The ability to adapt a trained model to another task is incredibly valuable.
This approach saves time and computational resources, making it more efficient to teach a computer to tackle new problems without making the model from scratch each time. This method is under-research so there is chance of getting more new concepts further. And also this is particularly useful when there is limited labelled data available for the new task, as the pre-trained model brings valuable general knowledge that can be fine-tuned for the specific requirements of the new task. It has proven to be highly effective in various applications.

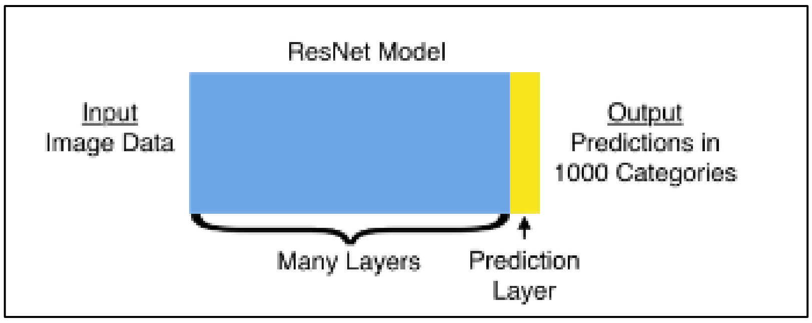


Fig. 2.18 Example of transfer learning

This is simple example of transfer learning where pre-trained model used is ResNet. Here the blue part is freeze because this is already trained with some large dataset(like available in ImageNet) and training should be done on only yellow part**.** This part is attached to the blue part. So there is a reason why time is saved in this method.
There are some method to achieve the transfer learning concepts in any project that is Fine tuning and Feature Extraction. These are two key processes in transfer learning, a technique widely used in deep learning. Let's explore each concept.

A. Feature Extraction

This method uses the features learned by a pre-trained network as input to a new classifier, which is trained from scratch. In this method using a pre-trained model to capture and extract relevant features from input data without modifying the model's weights. The lower layers of a pre-trained model, trained on a large dataset for a specific task, can serve as powerful feature extractors(or means removed the last fully connect layer). These layers have learned to recognize general patterns and features in the input data.

During feature extraction, the layers of the pre-trained model are typically **frozen,** meaning their weights are not updated during training. Only the additional layers for the target task are trained. In Fig. 19**,** the "top portion" of the model it the fully connected layer (green) and the pre-trained layers is called conventional layers.

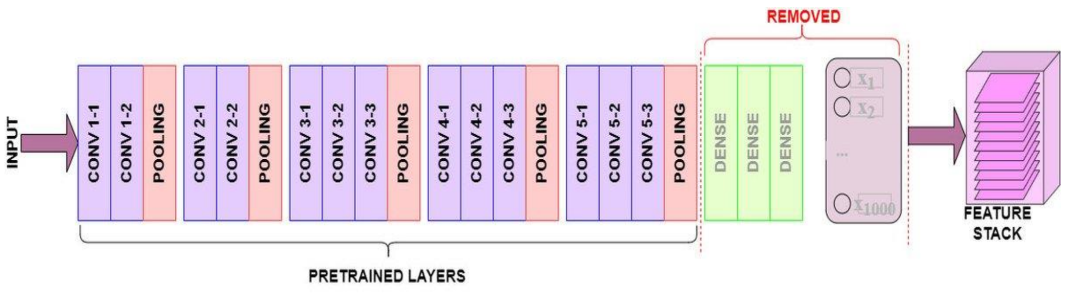


Fig. 2.19 Feature Extraction Example

The above graphic shows the "top" portion of the model as removed. The model convolves the pre-trained output layers left behind as a 3D stack of features maps. Freeze the convolutional layers and replace with the last dense fully connected layer. This method helpful when out own image base is kind of similar to ImageNet classification data-base.

B. Fine Tuning Approach

The primary objective of fine-tuning is to retain and adjust a portion of a pre-trained model. When fine-tuning a model, it involves further training on a new task while allowing some or all of its layers to be updated. This is distinct from feature extraction, where we only extract useful features. Fine-tuning adjusts the weights of specific layers in the pre-trained model to better suit the specifics of the target task. In fine-tuning, one or more layers of the pre-trained model are unfrozen, and their weights are updated during training, enabling the model to adapt to the subtle differences of the new task.

Fine-tuning is especially useful when the new task has a substantial amount of labeled data. It allows the model to adjust its parameters and learn task-specific pattern while retaining the knowledge gained during pre-training.

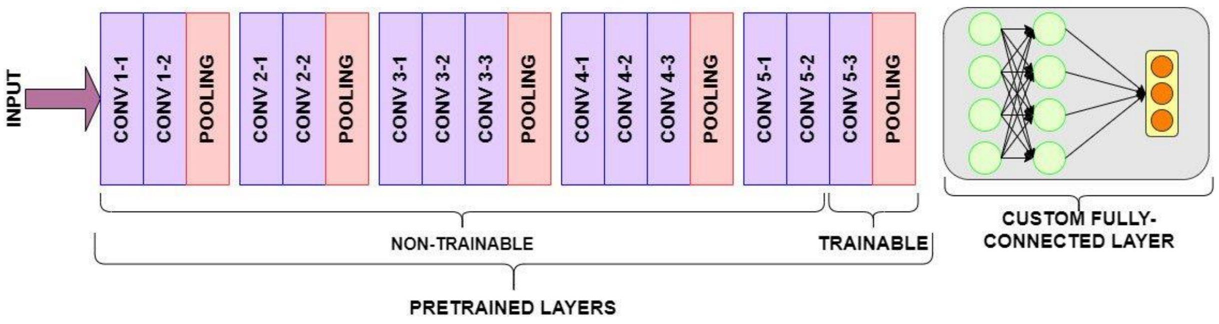


Fig. 2.20 Fine-Tuning Example

In the above graphic representation, the final convolutional and pooling layers are unfrozen to allow training. A Fully-Connected layer is defined for training and prediction.

3 Estimate Crack Features

Estimating crack features involves the process of analysing and identifying characteristics related to cracks in materials or structures. The task often includes using image processing, crack segmentation or data analysis techniques to quantify aspects such as crack length, crack width, pattern, crack orientation, and relevant properties. The goal is to obtain accurate measurements and descriptions of cracks, which is crucial in various field, including civil engineering, materials science, and non-destructive testing. The application of this approach plays a significant role crack monitoring of infrastructures , such as bridges and tunnel.

Guoliang et al. investigated types of crack on the surfaces and attempted to classify the cracks based on crack pattern. Therefore, it is increasingly import to extract and analysed crack patterns on the structural elements. The paper by Asjodi, Daeizadeh, Hamidi and Dolatshahi (2020) refer that nowadays, codes recommend manned inspections for monitoring of cracks. Since the manned inspection is prone to significant errors for monitoring cracks and the accuracy of this method depends on the experience and knowledge of the inspectors, many researchers have focused on computer vision methods to automate crack detection procedures.

There are some method to estimate the crack features that is Arc Length method or OTSU's method in digital processing in which crack length and crack orientation is figured out by regionprops function. In Arc Length method is introduced for extracting crack pattern characteristics, including crack width and crack length. The method contain two major steps; in the first step, the crack zones are estimated in the whole image that done in above using CNN and afterwards, the algorithm finds the start point , follows the crack pattern, and measures the crack features, such as crack width, crack length, and crack pattern angle. The paper by "Asjodi, Daeizadeh, Hamidia and Dolatshahi (2020)".

3.1 Image Segmentation

Image segmentation is a computer vision task that involves dividing an image into meaningful and homogenous segments or regions. To segment the crack it is necessary to create mask image of original image where segmented image is shown. In Fig. 3.1 is shown the sample example of crack segmentation. There are some important steps to achieve crack segmentation that is thresholding image by Otsu method, edge detection by Sobel. The two different framework are chosen for image segmentation purpose. The two different frameworks are UNet and LinkNet.

A. UNet Framework

UNet, evolved from the traditional CNN, was first designed and applied in 2015 to process biomedical images. It was originally presented by Olaf Ronneberger et al (2015). As a general CNN, focuses its task on image classification, where input is an image and output are one label, but in biomedical cases, it requires us only to distinguish whether there is a disease, but also to localize and distinguish. UNet is dedicated to solving this problem. The reason it is able to localize and distinguish borders is by doing classification on every pixel, so the input and output share the same size.

B. LinkNet Framework

It was presented originally by Chaurasia, Culurciello (2017). The purpose of the LinkNet was to perform real-time semantic segmentation of the image. LinkNet follow an encoder-decoder architecture, which is a common design in semantic segmentation models.

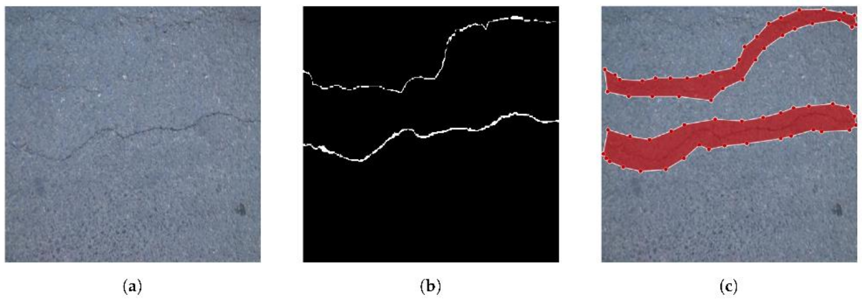


Fig. 3.1 Example of Crack segmentation (a) Original image; (b) Highlighted crack (Threshold) (c) annotated cracks with the segmentation masks.

3.2 Arc Length Method

The Arc Length method is crafted to identify a collection of dark pixels representing the lowest intensity within the crack flow. The distinguishing factor between the crack and other elements in the image lies in the sudden transitions occurring at the edge. This inventive approach employs several arcs to locate the positions of crack stations while simultaneously quantifying key characteristics of the crack, including the peak crack width and crack length.

In Fig. 3.2 the procedure of Arc Length method is graphically illustrated taken from the paper "Asjodi, Daeizadeh, Hamidia and Dolatshahi (2020)". To extract the crack pattern, the method first automatically searches for the start point of the cracks by circling around the boundary of a distinguished crack zone as shown in Figure 6b. If the crack is not found on the boundary, the zone is cut in half, and the searching is continued until the start point of the crack is found. Subsequently, the crack pattern is followed using Arc Length method by drawing different-sized arcs as shown in Fig. 3.2c.

Theoretically, the Arc Length method uses the intensity signal of R radius arc centered at the crack station points. The 1-D intensity diagram indicates the arc points' intensity variation against different angles. It should be mentioned that the grayscale image ranges from 0 to 255. Equation 1 presents a mathematical mapping which scales the darkest intensity into unity and the brightest into zero:

$$Color\ Index = 1 - \frac{pixels\ intensity}{225}$$

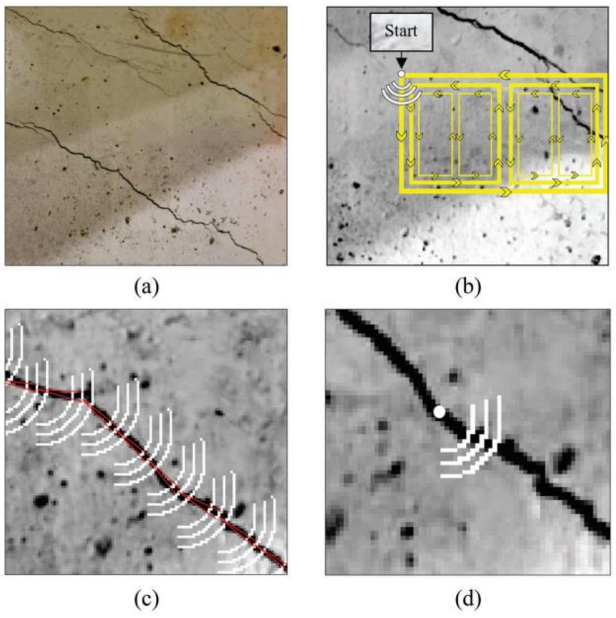


Fig 3.2 Arc Length method implementation: (a) original image, (b) start point detection, (c) crack flow recognition, (d) sample crack station.

3.3 Edge Detection

In the domain of crack detection, edge detection stands out as a pivotal technique for unveiling the intricate details of cracks within images. The traditional approaches like gradient-based methods, Laplacian of Gaussian (LoG), and the renowned Canny Edge Detector, have been extensively employed in crack detection applications. The adoption of edge detection techniques in crack detection not only contributes to the foundational understanding of crack characteristics but also serves as a crucial preprocessing step for subsequent analyses, laying the groundwork for advancements in structural health monitoring and maintenance practices.

There are some widely use edge detection methods:

A. Sobel operator :

The Sobel operator is a gradient-based method that calculates the first derivative of the image to emphasize regions of significant intensity change. The Sobel operator is effective in capturing vertical and horizontal edges, making it suitable for revealing linear features like cracks.

B. Canny Edge operator :

The Canny Edge Detector is a multi-stage algorithm that aims to detect edges while suppressing noise. It involves smoothing the image with a Gaussian filter, calculating the gradient to find edge strength and direction, performing non-maximum suppression, and applying hysteresis to trace and connect edges. Canny's ability to suppress noise and accurately trace edges makes it valuable in discerning the intricate boundaries of cracks.

4 References

[1] A. H. Asjodi, M. J. Daeizadeh, M. Hamidia, K. M. Dolatshahi, “Arc Length method for extraction crack pattern characteristics”, *Struct Control Health Monit,* (September 2020), DOI: 10.1002/stc.2653

[2] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for image recognition”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* PP: 770-778 (2016)

[3] K. Nguyen, C. Fookes, A. Ross and S. Sridharan, “Iris Recognition with Off-the-Shelf CNN Features: A Deep Learning Perspective” , *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* (2017), DOI: 10.1109/ACCESS.2017.2784352

[4] Özgenel, Çağlar Fırat (2019), “Concrete Crack Images for Classification”, *Mendeley Data*, V2, DOI: 10.17632/5y9wdsg2zt.2

[5] S. Dorafshan, R. J. Thomas and M. Maguire, “Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete” *Construction and building material,* Vol: 186, PP:1031-1045 (2018),
DOI: https://doi.org/10.1016/j.conbuildmat.2018.08.011

[6] Y. Cha and W. Choi, “Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks” , *Computer-Aided Civil and Infrastructure Engineering*, Vol: 32, PP: 361–378 (2017), DOI: 10.1111/mice.12263