Title: Development of Deep Learning Based Nonwoven Uniformity Analysis

Authors:

Eunkyoung Shim	PI
Mengmeng Zhu	Co-PI
Mingwei Gao	Graduate Student

Project Number: 22-263

Industrial Advisors: TBA – New Project

No. of Contacts: 0- New Project

Date Initiated: August 2024 **Date of this Report:** October 2024

ABSTRACT

This phase of the study reviewed key milestone models in the development of convolutional neural networks (CNNs) in the fields of image processing and object detection. The review covers the evolution from the Neocognitron model in 1980, inspired by animal vision, to AlexNet in 2012, which marked the commercialization of CNNs, and then to ResNet, which enabled the training of significantly deeper networks. It also includes more recent models such as DenseNet, MobileNet, Deformable convolutional networks, and EfficientNet, each contributing architectural innovations to progressively enhance CNN performance and computational efficiency . These models have laid a foundation for the application of deep learning in tasks like image classification and object recognition. This phase of the research summarizes these milestone models and their contributions, providing the theoretical basis for further development of CNN-based methods for nonwoven fabric defect detection and constructing a uniformity evaluation framework.

EXECUTIVE SUMMARY

The rapid advancement of artificial intelligence (AI) and machine learning (ML), particularly deep learning (DL), has brought revolutionary changes to fields like nonwoven fabric research. The uniformity of nonwoven fabrics is critical to their aesthetic and physical properties, such as permeability and tensile strength, and is key to improving product quality and production efficiency. In this field, one of the key applications of AI is computer vision-based image processing, especially using DL methods for defect detection and uniformity analysis. This study focuses on spunbond nonwovens, aiming to build a framework for uniformity analysis that can be expanded to other nonwoven processes in the future.

This report primarily focuses on the first task: the Literature Review. The literature review investigates convolutional neural networks (CNNs), which have emerged as a leading technique in DL-based image processing. Over the past few decades, CNNs have evolved significantly and have become a cornerstone method for tasks such as image classification, object detection, and other vision-related applications. Given the complexity of nonwoven fabrics and the various structural and defect characteristics that need to be analyzed, CNNs offer a robust solution due to their ability to automatically learn and extract relevant features from large datasets without manual intervention.

This report reviews the milestone papers in the development of CNNs, starting from the earliest work in 1980, Neocognitron, which was the first computer vision model inspired by animal vision principles. Then came LeNet, proposed by LeCun, which laid the foundation for modern CNNs. In 2012, AlexNet was introduced, marking a groundbreaking moment by transitioning CNNs from academic models to commercial applications. After that, CNNs evolved into deeper networks such as VGGNet and GoogLeNet, which improved performance by exploring deeper architectures. In 2015, ResNet was introduced, solving the vanishing gradient problem in deep networks and introducing residual connections, which made training much deeper networks possible. The report also covers more recent developments such as DenseNet, EfficientNet, and Faster R-CNN, which form the foundation for current popular architectures in image classification and object detection. Many of the latest studies build upon these methods for further development and application in various scenarios.

In conclusion, the report outlines the key development milestones of CNNs, laying a theoretical foundation for developing DL models suited for nonwoven fabric uniformity analysis.

1 Overview of CNN

1.1 Basic Introduction of CNN

In the field of DL, convolutional neural networks (CNNs) are one of the most well-known and commonly used algorithms [35][36]. Their main strength is their ability to automatically identify important features in data. CNNs are now widely used in many areas, including image classification, object detection, speech processing, and facial recognition. The structure of CNNs is inspired by neurons in the brains of humans and animals. In early research, CNNs were designed to mimic the way a cat's brain recognizes objects through its visual cortex [37]. The network has two main layers: one for identifying larger, rough features and another for picking out smaller, finer details [7].

Since Kunihiko Fukushima introduced the Neocognitron model in 1980 [7], CNNs have evolved and become a core tool for solving computer vision tasks. Unlike traditional neural networks, which often rely on manually designed feature extractors, CNNs can automatically learn features from data. This automatic feature extraction, particularly through convolution operations, makes CNNs highly efficient at processing large-scale image data.

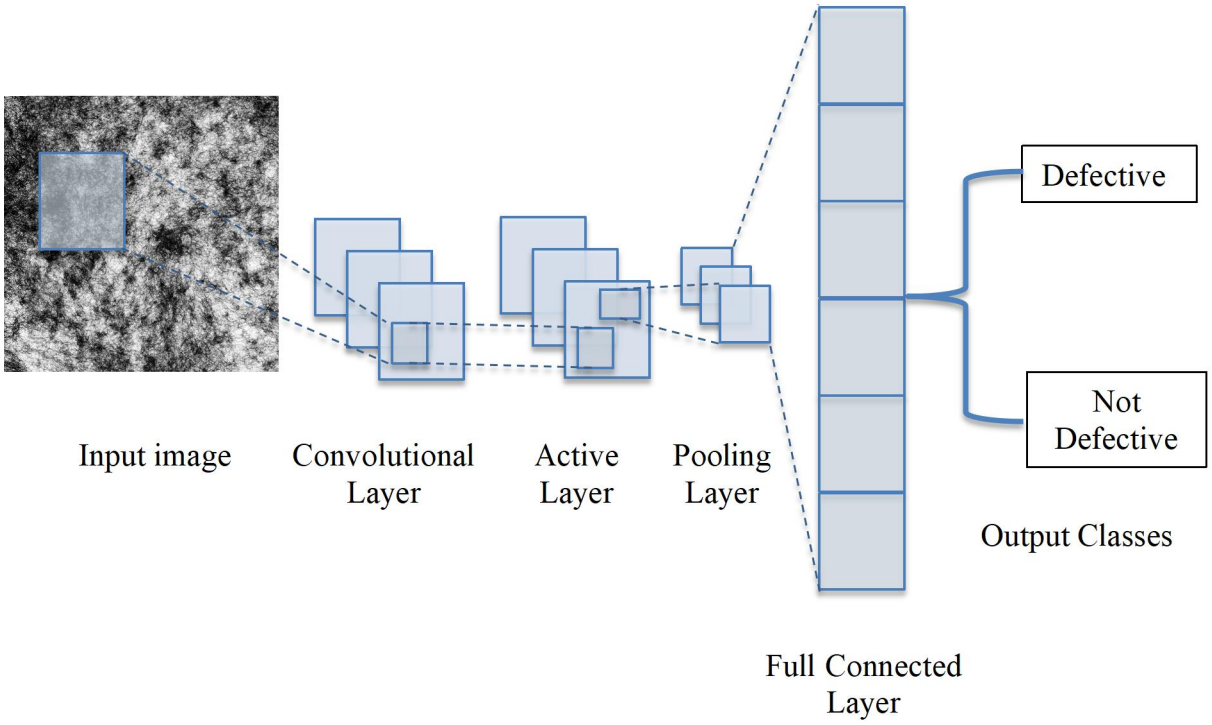


Figure 1: Using CNN for defect detection in nonwoven fabric.

This diagram illustrates the basic process of using CNNsto detect surface uniformity and defects in nonwoven materials. The image is first input into the network in grayscale, representing the microstructure of the nonwoven material. Then, the CNNs processes the

image through the following key steps. This report uses this diagram to explain the Basic Components of CNNs in detail.

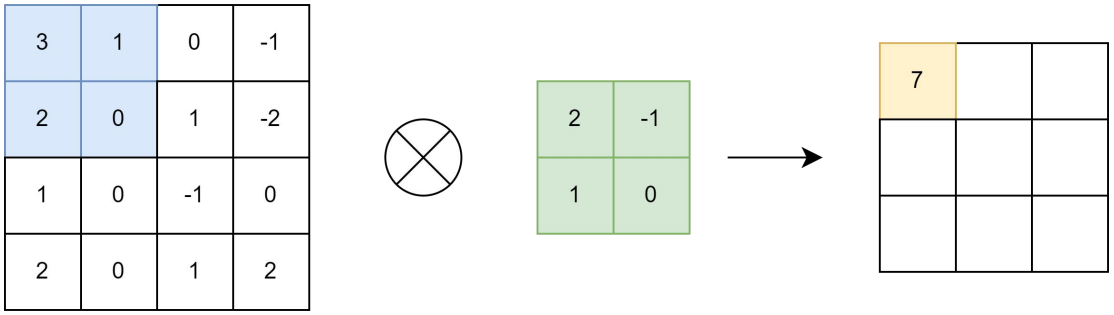
1.2 Basic Components of CNN

1.2.1 Convolutional layer

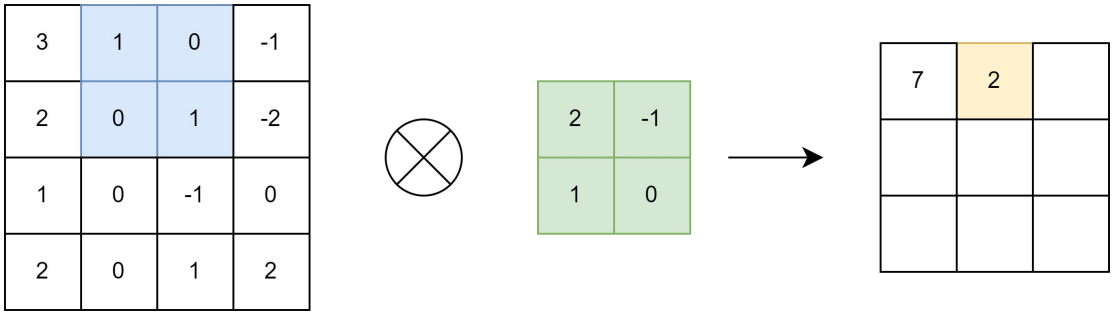
The image first passes through the convolutional layer, which is one of the most important components of a CNN [4]. It is made up of a set of convolution filters (also called kernels). The filters scan across the entire grayscale image to capture local features like the arrangement of fibers, uneven density, or the presence of defects.

Firstly, for the input format in CNNs, in traditional neural networks, inputs are in vector form, but CNNs use multi-channel images. For example, a single-channel input would be a grayscale image, while an RGB image has three channels.

Step 1



Step 2



Step 3

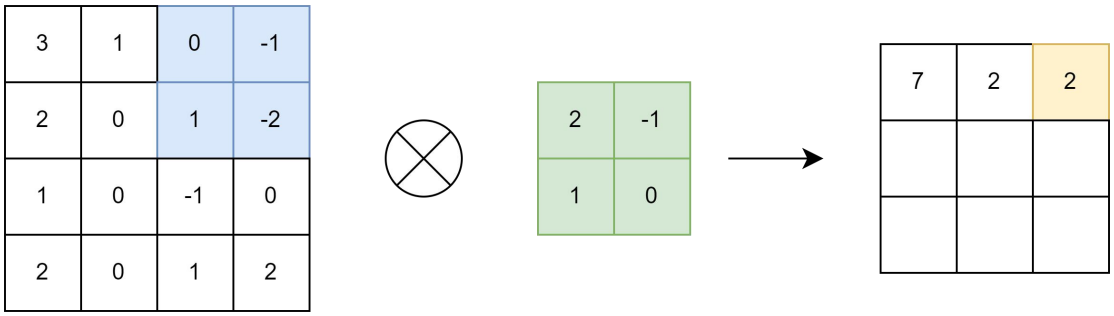


Figure 2: Illustration of convolution operation step-by-step.

The image is scanned by a filter (kernel) that has been initialized with random weights. The kernel slides horizontally and vertically over the entire image. At each position, the kernel and the corresponding part of the image are multiplied element by element (like matching pixels), and these values are then added together to create a single number, known as a scalar value. This process continues across the image, producing what's called a "feature map," which represents key details from the image.

The image is represented as matrix I and the kernel as matrix K. F represents the feature map value and F(x, y) represent the feature value at position (x, y). The convolution operation can be mathematically expressed as:

$$F(x,y) = \sum_i \sum_j I(x+i, y+j) \cdot K(x,y) \quad (1)$$

This formula shows that each value in the feature map is the result of multiplying corresponding values from the image and the kernel, then summing them up.

1.2.2 Pooling layer

The pooling layer plays an important role in reducing the size of the feature maps generated from the convolutional layer [9]. In simpler terms, pooling "downsamples" large feature maps to create smaller ones, while still preserving most of the important information (or features) from the image. This helps make the CNN more efficient and less sensitive to small changes or distortions in the input, such as slight shifts in the image.

There are several types of pooling methods, depending on the needs of the network[40]. These include tree pooling [38], gated pooling [39], average pooling [4], minimum pooling, maximum pooling [4], and global pooling methods like Global Average Pooling (GAP) [2] and Global Maximum Pooling (GMP) [41]. Among these, the most common ones used in practice are maximum pooling, minimum pooling, and GAP.

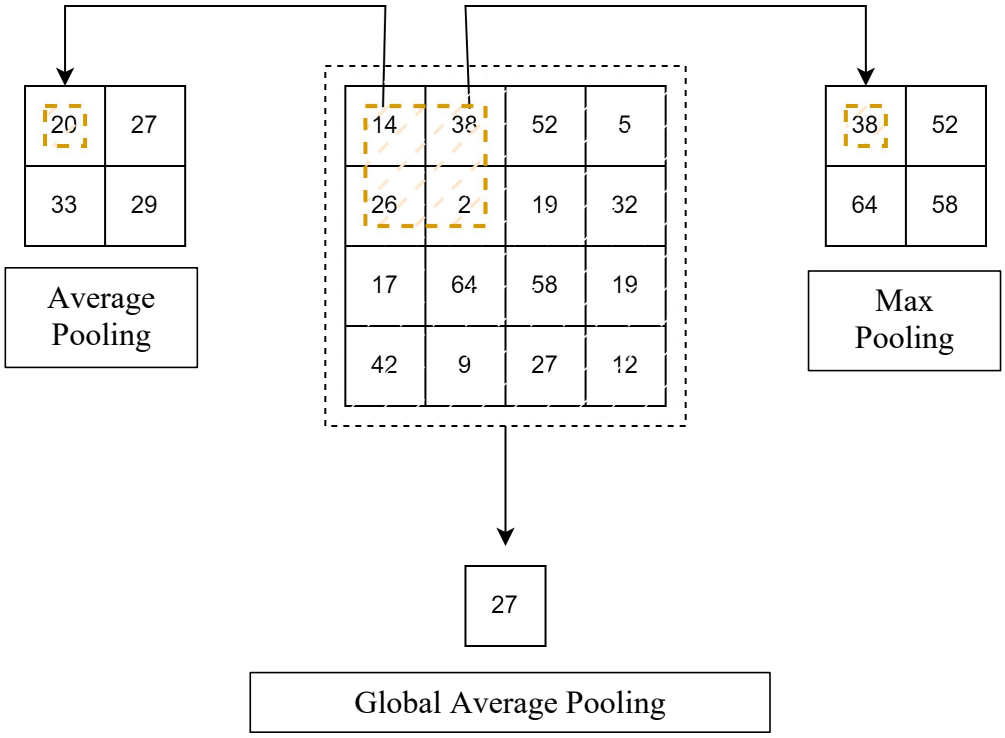


Figure 3: Comparison of average pooling, max pooling, and global average pooling.

In the context of nonwoven materials, maximum pooling can be particularly useful. For example, if the CNN is looking for defects like tears or unevenness in a nonwoven fabric, maximum pooling would focus on the strongest features (like the most prominent defect) in a specific area of the image. This way, even after reducing the size of the feature map, the key details about the defect remain intact.

1.2.3 Activation functions

Activation functions (non-linear) play a critical role in all types of neural networks by mapping inputs to outputs [42]. The main purpose of an activation function is to decide whether a neuron should be activated or not, based on the weighted sum of the input values and biases (if applicable).

In CNN architectures, a non-linear activation layer is applied after all weighted layers (also known as learnable layers, like fully connected layers and convolutional layers). The non-linearity of the activation function is important because it ensures that the mapping from input to output is non-linear. This non-linear behavior enables the CNN to learn extremely complex patterns in the data. Additionally, activation functions need to be differentiable, which is crucial for training the network using backpropagation [43].

The most commonly used activation functions in CNNs include ReLU (Rectified Linear Unit), which is efficient and helps avoid the vanishing gradient problem by outputting the input directly if it's positive and zero otherwise.

$$f(x)_{\mathrm{Re}LU} = \max(0, x) \tag{2}$$

A variation, Leaky ReLU, allows a small, non-zero output for negative values, preventing neurons from becoming inactive.

$$f(x)_{Leaky\,\mathrm{Re}LU} = \begin{cases} x & if \quad x > 0 \\ mx & if \quad x \le 0 \end{cases} \tag{3}$$

m represents the leakage factor, which is typically set to a very small value, such as 0.001.

The Sigmoid function maps input values between 0 and 1, making it suitable for binary classification tasks, though it is less commonly used in deeper networks due to vanishing gradient issues.

$$f(x)_{sigmoid} = \frac{1}{1+e^{-x}} \tag{4}$$

Finally, the Tanh function, which maps inputs between -1 and 1, is useful when both positive and negative values carry meaningful information, although it also shares similar limitations with Sigmoid.

$$f(x)_{Tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

1.2.4 Full connected layer

The fully connected layer is usually found at the end of a CNN. In this layer, every single neuron is connected to all the neurons in the previous layer. This is why it's called "fully connected." The main job of this layer is to take everything the CNN has learned from the image and make the final decision, such as which category the image belongs to [9].

Think of the fully connected layer as the final step in putting together all the puzzle pieces that the previous layers have found. Earlier layers might focus on detecting edges, textures, or patterns in an image, but it's the fully connected layer that takes all these pieces and says, "This is what the image represents."

The input to the fully connected layer comes from the last pooling or convolutional layer. This input is in vector form, created by flattening the feature maps generated from the previous layers. The fully connected layer processes this vector to make final predictions or classifications. It transforms the high-dimensional features into the desired output, such as class labels or probabilities, which represent the final result of the CNN.

For example, in nonwoven fabric defect detection, the fully connected layer would take all the features detected in the fabric and decide whether the fabric is "defective" or "non-defective."

1.2.5 Loss function and backpropagation algorithm

To know how well the CNN is performing, we need to calculate the error between the predicted output and the actual output. This is where the loss function comes in. The loss function measures the difference (or error) between the CNN's predicted output and the true output (labels) [44]. This error tells us how far off the model's predictions are, and it guides the CNN's learning process.

The loss function uses two key inputs to compute the error: the predicted output (the model's guess) and the actual output (the correct answer) [45]. By comparing these two, the loss function calculates how much the model needs to adjust its weights. Different types of problems require different types of loss functions.

Once the loss is calculated, the backpropagation algorithm kicks in. This algorithm helps the CNN learn by adjusting its weights. Backpropagation works by calculating the gradient of the loss function with respect to each weight in the network. Then, it updates the weights to minimize the loss in future predictions. You can think of this as the network "learning from its mistakes" — each time it makes an error, it updates itself to improve in the next round [9].

2 Architectures of CNNs

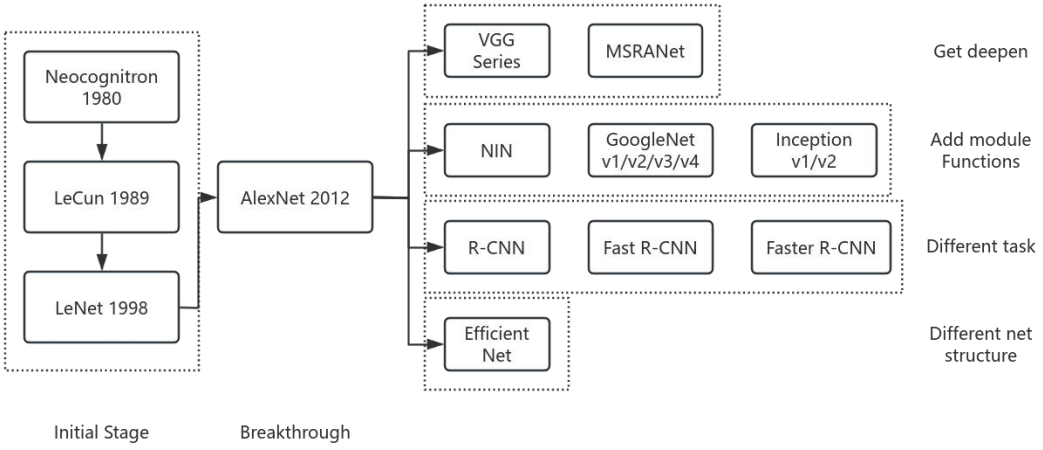


Figure 4: Evolution of CNNs: key milestones and model developments.

This diagram illustrates the key milestones in the development of CNNs. It begins with the early models, Neocognitron and LeNet, progresses through the breakthrough of AlexNet in 2012, and continues to more advanced models like the VGG series,

GoogleNet, Inception, Faster R-CNN, and EfficientNet. Each new model contributed by either deepening the network structure, adding functional modules, or optimizing for specific tasks, advancing CNN applications in image processing and object detection.

2.1 Early CNN Architectures

2.1.1 Neocognitron

Neocognitron, introduced by Fukushima et al. [7] in 1980, is considered the first model similar to modern CNNs. This model used multiple layers, each with its own specific job, to recognize images. The idea was inspired by how a cat's brain processes visual information, where different parts of the brain focus on different aspects of what the cat sees.

As shows in Figure 5, the Neocognitron had layers of "simple"(U_s) and "complex"(U_c) cells that worked together to identify patterns, making it a foundational step toward today's CNNs. The simple cells (U_{s1}, U_{s2}...) receive input from the previous layer and extract local features, such as edges and lines. The complex cells (U_{c1}, U_{c2}...) integrate the outputs from multiple simple cells to process higher-level features and provide robustness, allowing the network to recognize features across different positions and scales.

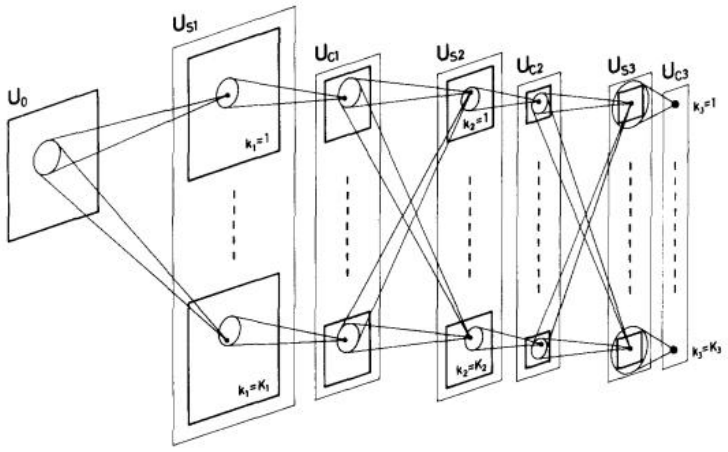


Figure 5: Structure of the neocognitron model with hierarchical layers [7].

2.1.2 Backpropagation algorithm

The backpropagation algorithm was introduced by Rumelhart and Hinton [8] in 1986. This method made training neural networks much easier and more practical. This breakthrough is a key part of how modern neural networks, like CNNs, are trained. Thanks to backpropagation, DL techniques have been able to grow and advance rapidly. According to Google Scholar, their influential paper has been cited 39,390 times, showing just how important their work has been in shaping the field of artificial intelligence.

2.1.3 LeNet network

In 1989, LeCun applied the backpropagation algorithm [9] to train a multi-layer neural network that could recognize handwritten postal codes. This work laid the foundation for what we now call CNNs. From this point on, the main structure of CNNs was set, including important parts like convolutional layers, pooling layers, and fully connected layers. Because of his contributions, LeCun is often called the "Father of CNNs." In his research, he used 5x5 filters in parts of the network to focus on small sections of the image, but at the time, he didn't specifically mention the term "convolution" or "CNNs".

It wasn't until 1998, when LeCun introduced LeNet-5 [4], that CNNs as we know them today were fully developed. Around the same time, the famous MNIST dataset, which is often used to teach machine learning, was also created. This dataset contains thousands of images of handwritten digits, and it became a standard for training and testing CNNs.

However, the LeNet model didn't become widely popular right away because computers at the time weren't powerful enough to handle it. Plus, other methods, such as support vector machines (SVM) [31], were able to achieve similar or even better results. Although LeNet was inspired by the earlier Neocognitron model, the Neocognitron didn't advance much. This was because LeNet used the backpropagation algorithm to fine-tune its settings, which made it more effective than the Neocognitron. As a result, the Neocognitron didn't see much further development.

It's important to note that in 2006, Bouvrie from MIT made a key contribution by explaining how to update the weights in CNNs more clearly [11]. This helped make the mathematical foundation of CNNs more standardized. Bouvrie's work also showed how the backpropagation algorithm could be applied more effectively to CNNs, adding more precision to the field and helping pave the way for the wider use of CNNs in the future.

2.1.4 AlexNet network

In 2012, AlexNet by Krizhevsky et al. [5] made a huge impact by winning the ImageNet competition with a big lead of 10.9 percentage points over the second-place team. This was a major breakthrough because it showed that CNNs could move from being just academic experiments to being used in real-world products. AlexNet is an 8-layer deep network, with 5 convolutional layers and 3 fully connected layers. It also introduced important features like ReLU activations, dropout to prevent overfitting, and GPU acceleration to speed up training, making it the first large-scale CNN to win a competition like this.

AlexNet introduced several key innovations that changed the way CNNs were used:

(1) ReLU Activation: AlexNet used ReLU (Rectified Linear Unit) as the activation function. Unlike previous functions like Sigmoid, which caused the "vanishing gradient problem" in deep networks, ReLU helped the model learn faster by keeping the gradient

strong in positive regions. This made training more efficient and faster with stochastic gradient descent (SGD).

(2) Dropout for Overfitting: Overfitting [34] occurs when a machine learning model performs very well on the training data but poorly on new, unseen data. This happens when the model becomes too complex, capturing not just the underlying patterns but also the noise in the training data, which reduces its ability to generalize to new data. To avoid overfitting, AlexNet used the Dropout method, which randomly turned off (or "dropped out") some neurons in the fully connected layers during training. This helped prevent the network from relying too much on any single neuron. About 50% of the neurons were dropped out in AlexNet's first two fully connected layers, which greatly improved its ability to generalize to new data.

(3) Overlapping Max-Pooling: Instead of using average pooling (which often blurred important features), AlexNet used overlapping max-pooling. This method kept more of the important details in the images, helping the network learn better.

(4) Local Response Normalization (LRN): AlexNet introduced LRN, which allowed neurons to "compete" with each other. Neurons that responded more strongly to features became more dominant, while weaker neurons were suppressed. This competition helped the model capture local details more effectively.

(5) GPU Acceleration: By using GPU acceleration, AlexNet significantly sped up training, making it possible to train larger and more complex networks.

(6) Data Augmentation: AlexNet used a technique called data augmentation by randomly cropping 224x224 sections from the original 256x256 images, along with their flipped versions. This effectively increased the size of the dataset by 2048 times. This method helped reduce overfitting and improved the model's ability to generalize to new data.

2.1.5 ZFNet network

ZFNet by Zeiler et al. [12] won the 2013 ImageNet classification challenge. While it didn't introduce any major new ideas, it made important improvements through careful tuning of the network's parameters, leading to better performance than AlexNet. ZFNet changed the first convolutional layer by reducing the filter size from 11 to 7 and the stride from 4 to 2. It also adjusted the 3rd, 4th, and 5th layers to have 384, 384, and 256 filters, respectively. These small changes led to a significant improvement in results.

2.1.6 VGG Network

In 2014, Visual Geometry Group (VGG) at the University of Oxford proposed VGG-Nets [6]. They were the base for the first-place model in the localization task and the second-place model in the classification task at the 2014 ImageNet competition. VGG is like a deeper version of AlexNet, with multiple convolutional layers followed by fully connected layers. At the time, it was considered a very deep network. VGGNet studied

how the depth of a CNN affects its performance by stacking many small 3x3 convolutional filters and 2x2 max-pooling layers, resulting in CNNs with 16 to 19 layers.

VGGNet, as shown in Figure 6, has six different versions, labeled A to E, each getting deeper but without adding too many extra parameters.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
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**
maxpool					
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**
maxpool					
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6: VGGNet configurations: comparison of layer depths and structures [6].

VGGNet has several important features: It is made up of 5 convolutional blocks, each containing 2-3 convolutional layers. At the end of each block, a max-pooling layer reduces the size of the image. The number of filters in each block stays the same within that block but increases as the network goes deeper, following this pattern: 64-128-256-512-512. In general, deeper networks tend to perform better. The authors also found that the Local Response Normalization (LRN) [5] layer didn't have much effect on the network's performance.

2.2 GoogLeNet/Inception V1 to V4

2.2.1 Inception V1

In 2014, Google Inception Net (GoogLeNet Inception V1) by Szegedy et al. [3] was introduced in the ILSVRC competition, the same year as VGGNet, and won the competition by a large margin. This version, called Inception V1, stood out for achieving excellent performance while keeping the model's computational and parameter complexity low, with a top-5 error rate of 6.67%. Its success came from a new network structure called the Inception module, which allowed different convolution sizes to be used in each layer. This design balanced efficiency and accuracy and was inspired by the "Network in Network"[2] (NIN) concept.

GoogLeNet has several key features:

(1) It introduces the Inception structure, a "Network in Network" design, where each node is like its own mini-network.

(2) The network uses auxiliary loss units in the intermediate layers, with a total of 3 loss units. These help the network converge by allowing the lower layers to contribute to the learning process. In the paper, these auxiliary losses are weighted by 0.3 and added to the final loss to improve training.

(3) Instead of using fully connected layers at the end like AlexNet, GoogLeNet replaces them with global average pooling layers, which greatly reduces the number of parameters. This helps make the network much more efficient. While fully connected layers in AlexNet make up 90% of its parameters, GoogLeNet removes them without sacrificing accuracy, achieving 93.3% accuracy on ImageNet, and it is also faster than VGG.

2.2.2 Inception V2

Inception V2 by Ioffe and Szegedy [13] is an important upgrade from V1. One key improvement is the introduction of Batch Normalization, which standardizes the output of each layer, reducing the issue of input distribution shift. This significantly accelerates the model's training speed while also making deep networks more stable, preventing issues such as overfitting. Additionally, Inception V2 adopts convolution factorization, breaking down large convolutional filters into smaller ones. For instance, a 5x5 filter is factorized into two 3x3 filters. This technique not only reduces the number of parameters but also greatly lowers computational costs, making the network more efficient while maintaining accuracy.

2.2.3 Inception V3

Inception V3 by Szegedy et al. [14] builds on the optimizations introduced in V2. V3 uses the more advanced RMSProp optimizer, which is an algorithm that controls gradients more effectively and speeds up the training process. It is better suited for deep neural networks compared to standard gradient descent methods, helping the model converge faster to the global optimum. V3 also introduces the Label Smoothing technique, which prevents overfitting by reducing the model's overconfidence in specific

classes. Simply put, label smoothing helps the model avoid becoming too certain when predicting a particular class, improving its performance across different datasets. Additionally, V3 continues to use the Inception modules in its architecture but further optimizes the use of convolution factorization, allowing the network to efficiently handle larger-scale images while maintaining both accuracy and efficiency.

2.2.4 Inception V4

Inception V4 by Szegedy et al. [15] combines all the improvements from V2 and V3, while also introducing Residual Connections, similar to those used in ResNet. This type of connection allows signals to skip certain layers, helping to address the vanishing gradient problem that arises as the network becomes deeper. By using residual connections, Inception V4 is able to build deeper networks without facing the usual difficulties associated with training very deep models.

2.3 Residual Learning Network

2.3.1 ResNet network

In 2015, ResNet by Kaiming He et al. [16] dominated the ILSVRC and COCO competitions, winning first place. ResNet made a huge breakthrough in how neural networks are designed, by introducing the residual module, going beyond just stacking layers and became a major milestone in DL.

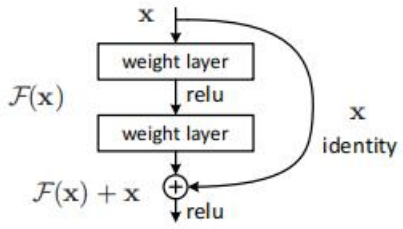


Figure 7: Residual learning block in ResNet [16].

The core idea of ResNet is the use of residual connections, as shown in the figure. These connections help solve the vanishing gradient problem, which often occurs when training very deep networks. Instead of letting the signal pass through every layer, a shortcut is added that lets the input bypass some layers and be added back to the output. This “skip” connection ensures that the network doesn’t lose important information as it gets deeper. The residual module, as shown in the figure, calculates the output of several layers (the function F(x)) and then adds the original input x back to this output. This simple addition prevents gradients from becoming too small during backpropagation, making it possible to train networks with over 100 layers, like ResNet-152, without the usual difficulties. This innovation has made deep networks more efficient and easier to train.

The left figure in Figure 12 shows a standard residual module with two 3×3 convolution layers. However, as the network gets deeper, this structure becomes less effective. To

solve this, the bottleneck residual block (shown on the right) gives better results. It uses three convolution layers: 1×1, 3×3, and 1×1. The 1×1 convolutions help reduce or expand the dimensions, allowing the 3×3 layer to work on lower-dimensional data, which makes the network more efficient.

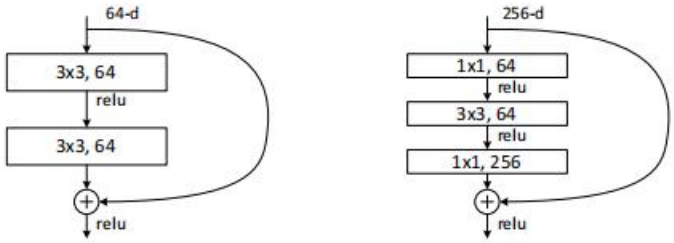


Figure 8: Comparison of standard and bottleneck residual blocks in ResNet [16].

2.3.2 DenseNet network

In 2017, DenseNet by Gao Huang et al. [20], won the Best Paper award at CVPR, though it didn't compete in the ILSVRC. DenseNet is fundamentally designed to solve issues like vanishing gradients and improve feature flow by using dense connections. Unlike ResNet or Inception, DenseNet connects each layer to all previous layers. This means that each layer's input comes not only from the previous layer but from the outputs of all preceding layers. This encourages feature reuse, as each layer can directly use features learned by all earlier layers. Additionally, this design improves information flow and significantly reduces the number of parameters, as it minimizes the need for large convolutional filters, making the network more efficient.

2.4 Region-Based CNN

The purpose of developing region-based CNN (R-CNN) was to address challenges in the field of object detection. Before DL was introduced into computer vision, traditional object detection methods relied heavily on handcrafted features (such as scale-invariant feature transform (SIFT) [46] and histogram of oriented gradients (HOG) [47]) and simple classifiers, such as SVM [31]). However, these methods performed poorly when dealing with complex and diverse images. The emergence of DL, especially CNNs, significantly improved the accuracy of image classification. Region-based CNN was developed based on this progress.

2.4.1 R-CNN

R-CNN by Girshick et al. [18] in 2014 is a DL model used for object detection. The key idea behind R-CNN is to first use selective search to generate multiple region proposals, which are candidate areas likely containing objects. Then, R-CNN processes each of these proposals individually using a CNN to extract features, which are then fed into a classifier for object detection. Although R-CNN improved detection accuracy, it was very

slow because it required running the CNN separately on each region proposal, leading to a large amount of computation.

2.4.2 Fast R-CNN

Fast R-CNN by Girshick et al. [19] in 2015 addresses the inefficiencies of R-CNN. Unlike R-CNN, Fast R-CNN runs the CNN only once on the entire image and then uses RoI pooling (Region of Interest pooling) to extract features for each region proposal from the shared feature map. This significantly reduces redundant computations. Additionally, Fast R-CNN performs both object classification and bounding box regression in a single network, further speeding up the process. In simple terms, Fast R-CNN improves detection speed and efficiency by sharing features and handling multiple tasks simultaneously.

2.4.3 Faster R-CNN

In 2015, Faster R-CNN was introduced by Ren et al. [17]. This DL model is designed for object detection, which is more complex than image classification. It faces two main challenges: handling a large number of possible object locations (candidate regions) and refining these regions to accurately pinpoint objects. Improving both the speed and accuracy of detecting objects is key to solving these challenges.

It works in four main stages. First, convolutional layers (conv+ReLU+pooling) are used to extract feature maps from the input image, which are shared across later layers. Next, the region proposal network (RPN) generates possible object regions by classifying anchor points as positive or negative using softmax, followed by bounding box regression to refine the proposals. In the third stage, RoI Pooling collects the feature maps and proposals, combining them to focus on the specific features of each proposal. Finally, the proposal feature maps are used to classify the objects, and bounding box regression is applied again to refine the positions of the final detection boxes.

Actually, the RPN places densely packed candidate anchors across the original image. A CNN is then used to classify these anchors as either positive (containing an object) or negative (not containing an object), making it essentially a binary classification task.

2.5 Deformable Convolutional Networks

2.5.1 Deformable convolutional networks v1

In 2017, deformable convolutional networks v1 (DCN v1) by Dai et al. [21]. from Microsoft Research Asia, was published at ICCV 2017. The key innovation of this model is the introduction of Deformable Convolution and Deformable RoI Pooling, which improve the network's ability to adapt to varying object shapes and sizes.

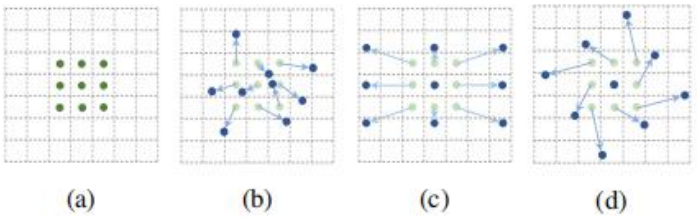


Figure 9: Sampling locations in standard vs. deformable convolutions [21].

In traditional convolution operations, the sampling points of the convolutional kernels are fixed (Figure 9(a)). For instance, a 3×3 convolution kernel always samples from fixed positions in a 3×3 grid. This works well for regular, geometric shapes. However, for more complex or irregular objects (like curved or non-standard shapes), this fixed sampling may not capture all the important features of the objects.

The key innovation in DCN v1 is the introduction of offsets, which allow the sampling points of the convolution kernels to be adjusted dynamically. Instead of using fixed grid locations, each kernel can shift its sampling points based on the input features. This makes the kernels more flexible and better at capturing the geometric structure of objects, improving the detection of irregularly shaped objects.

2.5.2 Deformable convolutional networks v2

In 2019, deformable convolutional networks v2 (DCN v2) [22] was introduced by Microsoft Research Asia and published at CVPR 2019. DCN v2 builds on the improvements of v1 by adding a learnable modulation scalar. This scalar allows the network to not only learn the offsets but also adjust the weights of the convolutional kernel at different positions. This makes the network even better at adapting to object shapes and improves its ability to extract important features.

2.6 Other lightweight or efficient networks

Since 2015, DL and CNN research have experienced rapid growth, with many new network models being developed. Researchers have explored various aspects, such as network architecture, functional modules, additional utility features, and different task applications. Below are some outstanding and representative papers from this period.

2.6.1 SqueezeNet

Squeeze-and-excitation (SE) by Hu et al. [30] works by first performing Squeeze, which captures global information from each channel by compressing it into a single value that summarizes the features. Then, in the excitation step, it adjusts the importance of each channel by reducing and increasing dimensions, allowing the network to give more weight to important features.

The design reduces computational complexity by splitting regular convolutions into two steps: in-channel convolution, which processes each channel independently, and cross-channel convolution, which combines information across channels. This two-step approach makes the computation more efficient while maintaining performance.

2.6.2 MobileNet V1/V2/V3

MobileNet V1 [29] is a lightweight CNN designed for mobile and embedded devices. Its key innovation is the use of depthwise separable convolutions, which split standard convolution into two steps: first, depthwise convolution processes each channel separately, and then pointwise convolution combines the outputs from different channels. This approach significantly reduces both the computational cost and the number of parameters, allowing MobileNet V1 to achieve high computational efficiency while maintaining good accuracy, making it highly suitable for resource-constrained devices like smartphones and IoT devices.

MobileNet V2 by Sandleret al. [28] builds on V1's foundation with further optimization by introducing inverted residuals and a linear bottleneck structure. Inverted residuals expand the channels, apply depthwise convolution, and then compress the channels back down, maintaining efficient feature flow while reducing computation. The linear bottleneck ensures that information loss is minimized by using linear activation in the final step of the residual block, avoiding the effects of non-linear activation.

Mobile inverted bottleneck convolution (MBConv) was introduced by the Google team in MobileNetV2 [28]. The Fused-MBConv layer was introduced later in EfficientNetV2 to further improve efficiency and performance.

MBConv consists of a 1x1 convolution, a depthwise separable convolution [29] (3x3), and another 1x1 convolution. It includes a SE [30] module to dynamically adjust channel weights.

MobileNet V3 by Howard et al. [48] combines the improvements from V1 and V2 with neural architecture search (NAS) and hand-designed modules to further enhance the network's performance. Key upgrades in MobileNet V3 include the SE module [30], which adaptively adjusts the importance of each channel to enhance feature representation, an optimized structure for the inverted residual block to make computation more efficient, and the replacement of ReLU with the Hard-Swish activation function, improving nonlinearity while reducing computational complexity.

2.6.1 EfficientNet V1/V2

In 2019, EfficientNet was introduced by Tan and Le [23]. The key innovation of EfficientNet is the compound scaling method, which scales three aspects of a CNN: network depth, network width, and input image resolution. Traditionally, networks are scaled by adjusting just one of these dimensions, like making the network deeper or

increasing the input resolution. However, EfficientNet improves performance by scaling all three dimensions together in a balanced way through compound scaling.

EfficientNetV2 by Tan and Le [27] builds on the original EfficientNet and introduces a new architecture that combines MBConv [28] and Fused-MBConv layers.

EfficientNetV2-S uses Fused-MBConv in the early layers to speed up training in the shallower part of the network. In the mid-to-deep layers, it switches to MBConv layers, which include the SE module to improve parameter efficiency and boost accuracy.

SUMMARY & CONCLUSIONS

In the literature review, a comprehensive overview of the milestone papers in the development of CNNs was conducted.

Table 1: Milestone Models in the Development of CNNs

Year	Model	Contribution
1980	Neocognitron [7]	The first computer vision model based on animal vision, laying the foundation for early CNN development.
1990s	LeNet [4][9]	Proposed by LeCun, it laid the foundation for modern CNNs
2012	AlexNet [5]	Marked a milestone in transitioning CNNs from academic models to commercial applications.
2013	ZFNet [12]	Improved performance over AlexNet by reduced the first convolutional layer's filter size and adjusted the number of filters in deeper layers
2014	VGGNet [6]	Improved image classification accuracy by using deeper network structures with smaller convolutional kernels.
2014	GoogLeNet [3][13][14][15]	Introduced the Inception module, balancing network depth and width while improving computational efficiency.
2015	ResNet [16]	Solved the vanishing gradient problem in deep networks with residual connections, enabling the training of much deeper networks, a milestone in deep learning.
2015	Faster R-CNN [17][18][19]	Introduced the RPN, greatly improving the accuracy and efficiency of object detection, becoming a standard architecture in this task.
2017	DenseNet [20]	Introduced dense connections, promoting feature reuse and reducing the number of parameters, further improving network efficiency.
2017	MobileNet [28][29][48]	Introduced depthwise separable convolution, significantly reducing model parameters and computational cost while maintaining accuracy.

2017	Deformable Convolutional Networks [21][22]	Introduced deformable convolutions, allowing convolution kernels to dynamically adjust sampling positions, improving the network's ability to detect irregular objects.
2019	EfficientNet [23][27][30]	Proposed compound scaling, optimizing the depth, width, and resolution of the network simultaneously for higher computational efficiency and performance.

In conclusion, this literature review lays the theoretical foundation for the subsequent development of DL models for defect detection and uniformity analysis in nonwoven fabrics.

FUTURE WORK

The initial subject and time planning for this project was as follows:

1. Literature review — until June 2025
   - Overview of image processing technology
   - Object recognition and defect detection in nonwoven fabrics

2. Image acquisition/organization
   Synthetic Hybrid image creation — until June 2025

3. Develop nonwoven defect/non-uniform feature detection and analysis model with CNNs: Initial model architecture development — Jan. - Dec. 2025

   CNN model - Training/testing model architectures & model performance evaluations, and model refinements — Sep. 2025 – June 2026

   Evaluate imaging conditions/image quality, nonwoven surface textures on defect/non-uniform feature analysis model performances — Sep. 2025 – June 2026

4. Identify potential algorithm for overall uniformity analysis — Sep. 2025 – Dec. 2026

   Initial overall uniformity models architecture development and feasibility test — Jan. - Dec. 2026

   Training/testing selected model architectures & model performance evaluations, and model refinements — Sep. 2026 – June 2027

   Evaluate imaging conditions/image quality, nonwoven surface textures on the overall uniformity models — Jan. - June 2027

According to the plan, the project is currently in the literature review phase. First, a comprehensive review of image processing methods will be conducted, with a focus on

CNNs and their derivatives, Recurrent Neural Networks (RNNs) and their derivatives, as well as Transformers and other popular methods used in image processing today. The goal is to categorize the applications of these methods and gain a thorough and clear understanding of image processing techniques within DL.

Next, the focus will be on the field of object recognition, particularly methods related to defect detection. This will lay a methodological foundation for future research into a nonwoven fabric uniformity detection framework. Finally, particular attention will be paid to current defect detection methods in the nonwoven fabric field, summarizing their strengths and weaknesses, and identifying the direction for developing a universal uniformity detection framework.

Simultaneously, during the literature review phase, the process of collecting nonwoven fabric images will be carried out in parallel. This includes obtaining real images from the production process, as well as generating synthetic images by superimposing defects. Ultimately, a complete dataset of nonwoven fabric images will be constructed to support the development and testing of DL models.

REFERENCES

[1] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable Convolutional Networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 764-773.

[2] Lin, M., Chen, Q., & Yan, S. (2014). Network In Network. arXiv preprint arXiv:1312.4400.

[3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9.

[4] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, 86(11), 2278-2324.

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25, 1097-1105.

[6] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[7] Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4), 193-202.

[8] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536.

[9] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 541-551.

[10] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. IEEE.

[11] Bouvrie, J. (2006). Notes on convolutional neural networks. Retrieved from https://web.mit.edu/jvb/www/papers/cnn_tutorial.pdf

[12] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Proceedings of Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Part I 13,* 818-833. Springer International Publishing.

[13] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 448-456.

[14] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818-2826.

[15] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4278-4284.

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

[17] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28, 91-99.

[18] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 580-587.

[19] Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1440-1448.

[20] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4700-4708.

[21] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 764-773.

[22] Zhu, X., Hu, H., Lin, S., & Dai, J. (2019). Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9308-9316.

[23] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 6105-6114.

[24] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1904-1916.

[25] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., & Feng, J. (2017). Dual path networks. *Advances in Neural Information Processing Systems*, 30.

[26] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1492-1500.

[27] Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning* (pp. 10096-10106). PMLR.

[28] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510-4520.

[29] Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[30] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132-7141.

[31] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

[32] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580-587).

[33] Girshick, R. (2015). Fast R-CNN. arXiv preprint arXiv:1504.08083.

[34] Box, G. E. P., & Tiao, G. C. (1973). *Bayesian Inference in Statistical Analysis*. Addison-Wesley.

[35] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354-377.

[36] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90.

[37] Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1), 106.

[38] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12, 2493-2537.

[39] Cho, K. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[40] Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10),* 111-118.

[41] Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2015). Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 685-694.

[42] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5, 115-133.

[43] Xu, B. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

[44] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386.

[45] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548.

[46] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 91-110.

[47] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886-893). IEEE.

[48] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314-1324.