Deep Convolutional Nets

Hendrik Pfaff, Victoria Chaikovska
Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering

***Abstract*—We introduce the basic concepts of Convolutional Neural Networks (CNN), their layered architecture and important hyperparameters. Their historic development since their neurophysiological real life inspiration get described. We also point out the differences between classical Multilayer Perceptrons in the aspects of interpreting images and learning while explaining the components, layers and hyperparameters of a CNN.**
We also explain the difficulties of training of very deep networks with Stochastic Gradient Descent. But the new method Batch Normalization helps significantly to improve the training of deep neural nets. We elucidate the relationship between the frequency spectrum of image data and the generalization behavior of convolutional neural networks. Deep convolutional neural networks are widely used in many different fields, such as robotics, speech recognition, sentence classification, medicines, economics and others.

***Index Terms*—Learning From Data; Convolutional Neural Networks; Machine Learning; training CNN; Stochastic Gradient Descent; Deep CNN; generalization; CNN for sentence classification; robot detection; crystallization outcomes; stock market prediction.**

I. Introduction

DEEP Convolutional Neural Networks (CNN) are a recent and broadly used technology in supervised machine learning. Like for many other machine learning techniques, the underlying biological concepts and components were researched as early as the 1960s. Numerous scientists like *Hubel* and *Wiesel* of in the field of biology and physiology took inspiration from the cells in the human visual cortex for their so called spatial invariance [1].

Based on this work, Dr. Kunihiko Fukushima introduced the concept on *Neocognition* in the 1982 [2]. This approach tried to mathematical formalize different layers of visual perception with two different types of cells.

This approach was developed further in the 1990 by *Yann LeCun et al.*, into what is now called CNN. Their proof of concept of a multilayer neural network was able to accurately recognizing handwritten digits from a MNIST Database [3].

Modern Deep CNN can be considered as a regularized derivative of classical Neural Networks (NN), which they improve by adding several connected dedicated layers in their architecture. This adjustment enables CNN to be invariant to shift, translation and space of input (SIANN) [4]. Due to this, the most common application for CNN is in the field of image analysis and recognition. Though use cases in other disciplines like text analysis or economics are also mentioned and shown in this report.

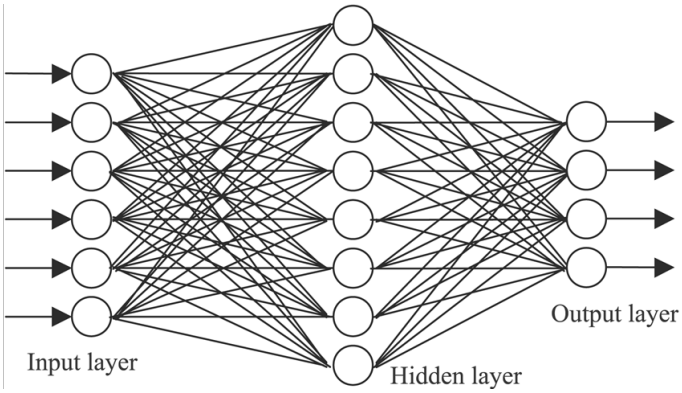

Fig. 1. A graphical representation of a classic artificial feedforward neural network. It consists of one *Input*, one *Hidden* and one *Output* layer with several nodes in each. [5]

This Report is intended to introduce the reader into the topic of Deep CNN. It focuses on the technical foundations of CNN like the architecture, layers and hyper parameters as well as the actual use and training of CNN.

This Paper is structured as follows. Section II begins with a brief description of the fundamentals of neural networks and their relevance for CNNs. Following this we explain the structure of CNN architecture in section III. Section IV goes into detail of the different layers of a CNN, their properties and parameters, while section V explains the most used hyperparameters *Filters*, *Stride* and *Zero-Padding*. Section VI describes how CNNs are trained. In section VII problems of overfitting and generalisation are discussed. The most used fields for applying CNNs are introduced in section VIII which is then followed by a conclusion in the final section IX.

II. Classical Neural Networks

As mentioned in the Introduction in section I, we explain the foundations of classical NN in this section to have a comparison with CNN it in the further report.

Neural networks consist of multiple layers of perceptron nodes and are thus called *Multilayer Perceptrons* (MLP). The functionality of a single perceptron is inspired by biological brain cells and their action potential firing. Similar to their biological counterparts, perceptrons receive input values that are weighted and a bias. These parts are add up and passed

to a activation function which generates the output value of the perceptron.

Several perceptrons can be included in a layer of which several layers can be structured in orderly manner after each other. Each output value of one perceptron is thus passed and weighted as input to each node of the following layer. The first layer, also called Input Layer, receives the features of the input data. The last layer, called the Output Layer, returns the result of the MLP (i.e. classification, regression).
To learn, the MLP needs to modify its weights for each input value in each layer. For this the error backpropagation process is used. It recalculates the weights backwards on basis of the right or wrong result in the training data set [6].

While MLP have a lot of useful applications, there are major disadvantages when it comes to the classification of images.

The number of features coming from a single image can be quickly very large. For greyscaled or black-and-white images, the greyscale value of every pixel is used as a feature. When using colored images, the number of color channels (i.e. for RGB, CMYK) multiplies with the image dimensions resulting in $width \times height \times \#\ of\ channels$ features.
MLP also aren't invariant to the translation or rotation of the objects within an image [4]. Similar pictures of the same object but in different positions would lead to completely different calculations of weights of the MLP.
Both of those problems are not easily resolved in real-world applications, where images are not always centered of evenly focused. Preprocessing or preparing those image data by hand is often too inefficient.

III. Architecture of CNNs

After we discussed the structure of classical known neural networks in section II, we now focus on the architecture of CNNs, their similarities and differences. Specifically the setup of different distinct layers and their connection to each other.

While the different layers of a MLP are often divided into input, hidden and output layer, their inner workings stays the same. Each node in each layer receives a weighted input and calculates the output with its activation function and passes its result.
In a CNN, layers differ in their procedure and purpose within the network. The first layer is, like in a MLP, the Input Layer to receive the data features. After this, a number of Convolutional Layers execute the convolution process with different filters to detect the different features of the image. The multi dimensional output of the convolution is usually added up and passed to the ReLU function. For dimensionality reduction, this output is committed to a Pooling Layer. The alternation between Convolution and Pooling Layers can happen for several times, depending on the performance of the CNN, as later appearances of Convolutional layers are able to extract higher level features as earlier. In the last layers of the CNN, the Flatten Layer reduces the dimensions of the input volume get even further as the Pooling Layer.
With one dimensional data, the Fully Connected Layer now finally classifies the image into labels, similar to a MLP, as described in section II.

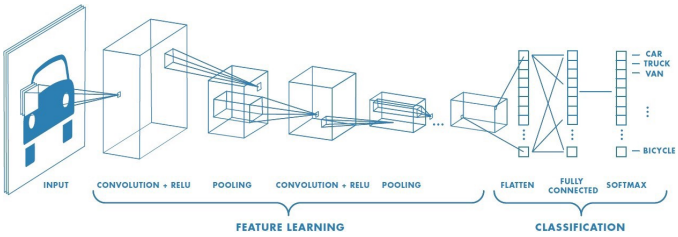

Fig. 2. Example architecture of a CNN processing an image. The input image is processed by a number of consecutive convolution and pooling layers before it can be classified and labeled. [7]

The key difference between a MLP and a CNN is the purpose of the dedicated layers between their Input and Output Layer.

IV. Layers

This section gives deeper insight into the different types of layers within CNNs, as mentioned in section III. We describe the Input Layer, the Convolution Layer, the Pooling Layer, the Flatten and the Fully Connected Layer. Each one of the aforementioned layers fulfills a certain purpose in the CNN.

A. Input Layer

The main task of the Input Layer is the extraction of features from the input image. As mentioned in section II, the number of features gained from images can be very large, depending on the type of image. The dimensions of the input volume is relevant to the whole training process of the CNN. A black and white image can be translated into a two dimensional, binary valued matrix with the dimensions of the image. Greyscaled pictures produce values according to the scale of the pixel $(0-255)$ in a two dimensional matrix. Colored images create a three dimensional input volume, which uses its color channels as third dimension (see Fig. 3).

B. Convolution Layer

The Convolution Layer is the eponymous Layer of the CNN. It performs the convolution process with a defined filter on its input volume to detect the different characteristics of an image. While the filters in early appearing Convolution Layers only extract low level characteristics like edges or blur, with increasing number of Convolution Layers, more high level characteristics and objects can be detected.

To calculate the convolution, it can be described as the

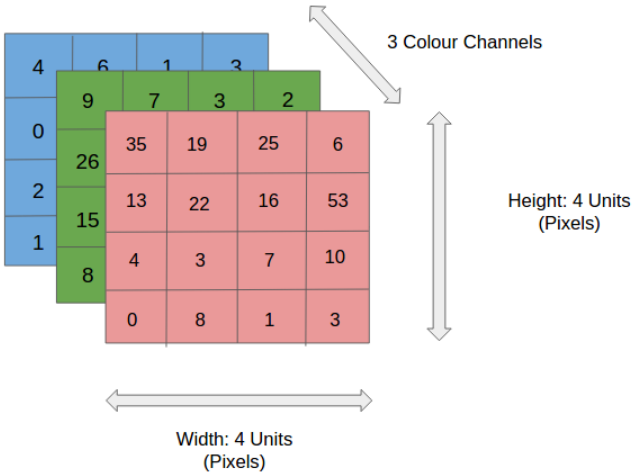

Fig. 3. Colored images translate into three-dimensional input volumes [7].

filter setup laying above the input image in the upper left corner. The values of the filter are multiplied with the underlying values of the image in the current position. These products are add up and result in the new convoluted output. Afterwards the filter moves along stepwises to its next position on the input.
The convolution itself depends on the setting of several hyperparameters. Section V will further explain the most important ones.

As important as the dimensions of the input volume, are the dimensions of the output after the convolution process. The size of the output volume, dependent on the hyperparameters, can be calculated by equation 1.

$$Output\ size = \frac{W - K + 2P}{S} + 1 \tag{1}$$

The parameters of the output function are the values of the respective hyperparameters. W equals the width or the height of input image. K equals the width or height of the applied filter for this Convolution Layer. P equals the set Padding around the input. S equals the stride of the filter.

The *rectified linear unit* (ReLU) function (eq. 2), passes its given input value as long as it is positive. Otherwise the value 0 is returned. It was first introduced in 2000 by Hahnloser et al. [8] and became one of the most defining functions for CNN [9].

$$ReLU(x) = max(x, 0) \tag{2}$$

The ReLU function is not differentiable in its origin. Thus it is often approximated in practice by the *Softplus* function (eq. 3) for the backpropagation calculation [10].

$$Softplus(x) = ln(1 + e^x) \tag{3}$$

C. Pooling Layer

In an image, neighboring pixels often have similar values and lead to similar results in the convolution process. This redundant information may increase the output size of the Convolution Layer, but is not necessarily important for classifying an object within the image. The main purpose of the Pooling Layer is the reduction of dimensions by a specified pooling size, after the convolution process. This process can be compared to subsampling or compression processes.
Similar to the movement of the filter in the Convolution Layer, a window of the pooling size. slides over the input and executes a mathematical operations over all underlying values. The result of this operation becomes the new smaller input for further layers. The most common operations used in the Pooling Layer are *max*, *min* and *avg*.

D. Flatten and Fully Connected Layer

The last layers of a CNN are the Flatten and Fully Connected Layer. The Flatten Layers purpose is to transform the output volume of all previous Convolution and Pooling Layers into a one dimensional layer of nodes [7]. These nodes work as input for the following Fully Connected Layer. The Fully Connected Layer finally classifies its input into the possible pre-defined classes.

The Softmax, or *Softargmax*, function originated from a 1868 work of austrian physicist *Ludwig Boltzmann* [11]. Yet, its application in modern machine learning techniques was first established in 1989 by John S. Bridle [12]. It is mostly used for the last classification layer of the CNN when the available output classes are mutually exclusive. [13]

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4}$$

When looking at the equation of the Softmax function (eq. 4), we can see how it is basically a normalized exponential function. The Softmax function turns its real input values into probability values that sum up to one in a probability distribution.
When applying the Softmax function to classify the output of the Fully Connected Layer, the output node with its highest value / highes probability becomes the label for the image.

Another possible activation function for the final classification is the Sigmoid function (eq. 5). This function has many applications and is often used activation function for MLP due to its easy implementation and differentiation properties. The Sigmoid function is usable for classification an image to multiple labels [13].

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

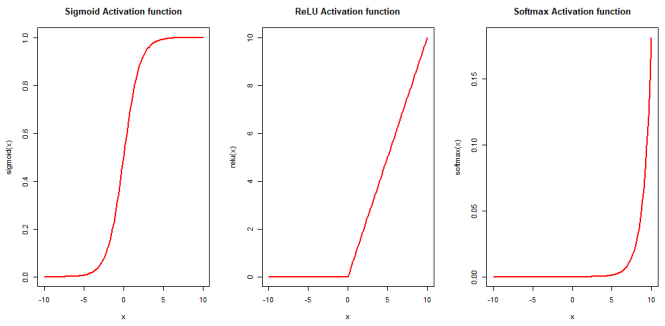

Fig. 4. Plot of the sigmoid, ReLU and softmax activation functions.

V. Hyperparameters

As mentioned in section IV, hyperparameters have a big influence in the quality of outcome of convolution. This section will explain the three of the most important hyperparameters for the convolution process, filters, stride and zero-padding.

A. Filter

The filter is the core of the convolution process. Often filters are quadratic, with an uneven side length (i.e. 3×3, 5×5 or 7×7). The filter values define its main purpose. Edge detecting filters often highlight certain fundamental features of an image while filters for blur may be used to increase the detection of higher level features [14].

B. Stride

The stride parameter defines how many pixels the filter is moving over the input in each step of the convolution process. While the standard value is 1, higher values result in a lower overlap and resolution of the filter.

C. Zero-Padding

Zero-Padding is defines an area around the outer pixels of the input image and fills it with the value 0. It is used to modify the size of output dimensions and prevent losing information from the outer borders of the image [15]. If no padding is defined, the size of the output volume will be smaller than the input volume, due to the nature of convolution. This is called *Valid Padding*.
To keep the size of input and output the same, the so called *Same Padding* can be applied. It adds p additional rows of 0 values around the input image. The value p can be calculated by equation 6.

$$p = \frac{filter\ size - 1}{2} \tag{6}$$

To increase the size of the output volume after convolution, a padding area up to the size of the filter can be applied, which is called *Full Padding*.

VI. Training CNNs

Training of Deep Neural Networks is a complex process because during the training changes of each layer input are distributed due to transformation of the parameters in the previous layers. All this decelerates the training requiring lower learning rates, therefore the training of models with nonlinearities is very difficult. In general, the CNN requires a large amount of data for training [16].

Because of the difficulties in deep network training, researchers have tried to elaborate the better one. Correctly designed initialization strategies for activation functions have been modified in order to improve the training process. This strategies presented good results by testing.It has been shown that the activation functions, which are based on local competition improve the training. In neural networks the connections between layers and to output layers (in case of injected error) are utilized with the goal to optimize the data flow. This approach uses soft targets from a superficial teacher network to support the training of deep student networks on multiple levels. The deep networks can be trained layer-wise in support of credit assignment, but this technique is not so effective like direct training [17].

A. Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is an efficient way of training deep network and it is utilized to reach state of the art performance.

Stochastic gradient descent optimizes the parameters Θ of the network, therefore the loss function minimizes

$$\Theta = \arg\min \frac{1}{N} \sum_{i=1}^{N} \iota(x_i, \Theta) \tag{7}$$

where $x1 \ldots n$ is a set of training data. The training with the Stochastic gradient descent approach is executed in steps, every step contemplate a mini-batch $x1 \ldots m$ of size m. For approximation of gradient of the loss function is used mini-batch by computing with the following parameters

$$\frac{1}{m} \frac{\partial \iota(x_i, \Theta)}{\partial \Theta} \tag{8}$$

Utilization of mini-batches is useful in many aspects. First of all it is important to clarify, that the gradient of the loss function over a mini-batch and an estimation of the gradient are beyond the training data set. The quality of gradient gets better while increasing batch size. Calculation over a batch is more effective than m calculations for individual models due to more powerful computing platforms. It needs time-consuming tuning of model hyper-parameters as well as the initial values such as the learning rate for parameters of the model. The training is complex because inputs to each layer depend on the parameters of all layers. It means, that a few changes by the parameters of network increases the probability that the network becomes deeper.

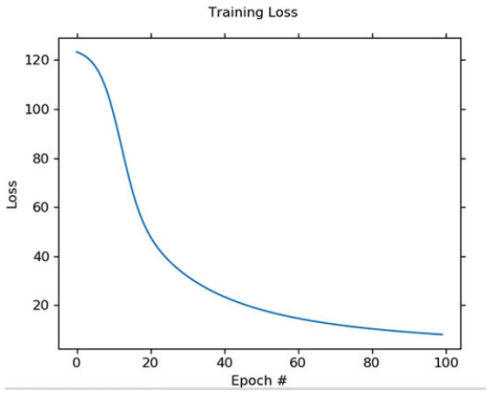

Fig. 5. Plotting loss over time using gradient descent.Plot demonstrates how the loss over time has decreased dramatically. As you can see we have initially very high loss, but during the training SGD Algorithm minimizes this loss [18]

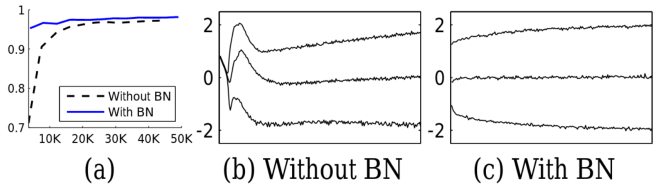

Fig. 6. The test accuracy of network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and reach higher accuracy. Images *(b)* and *(c)* show the evolution of input distributions to a sigmoid [16]

The change in the inputs in distributions of layers is problematical, the layers can not quickly adapt to the new distribution changes. By the changes in the input distribution to a learning system, it should learn covariate shift. This is normally carried out via domain adaptation. The idea of covariate shift can be extended over the learning system and operate with its parts, like a sub-networks, regarding the network calculation ι = *F2 (F1(u,*$\Theta 2), \Theta 1$*)*. F1 and F2 are random transformations and the parameters $\Theta 2, \Theta 1$ represent the ability to learn the minimization of the loss ι. Learning $\Theta 2, \Theta 1$ can be shown as if the inputs *x = F1(u,* $\Theta 1$*)* are passed to the sub-network ι = *F2(x,* $\Theta 1$*)*.

The gradient descent step, for example

$$\Theta 2 \leftarrow \Theta 2 - \frac{a}{m} \sum_{i=1}^{m} \frac{\partial F2(Xi, \Theta 2)}{\partial \Theta 2} \tag{9}$$

Batch size *m* and learning rate *a* correspond to a stand-alone network *F2* with input *x*. The input distribution properties make training more effective and contribute to training the sub-network. $\Theta 2$ does not have o be adjusted in order to balance the changes by the distribution of *x*. The distribution of inputs to a sub-network can have positive effects for the layers outside the sub-network. Regarding the layer with the sigmoid activation function:

$$z = g(Wu + b) \tag{10}$$

where u is the input, vector b is the layer parameters for learning, *g(x) = 1 + exp(1-x) n* and *W* is the weight matrix. Where *g(x)* tends to zero increasing of *X*.

This implies that for all dimensions of *x = W u + b* except only those with small absolute values the gradient converges to u and the the training of model slows down. Because of the influence from x by *W, b* and the parameters from all layers, parameters vary during the training and make the process slower.

This effect becomes better as the depth of network increases. In reality, the results of evaporate gradients are normally addressed by using Rectified Linear Units and low learning rates. When the distribution of nonlinearity inputs stays robust while the network is training the optimizer hangs in the satisfied mode, and it makes the training faster.

B. Batch Normalization improves learning rates

During the training such as Internal Covariate Shift the distributions of internal nodes of a deep network is changed. By deactivating it the training becomes faster. It makes sense to utilize a new approach called Batch Normalization, that helps to reduce internal covariate shift and significantly improves the training of deep neural nets. It achieves this state with a normalization step that fixes the differences of layer inputs. Batch Normalization has an advantageous impact on the gradient flow. It provides an opportunity to use much higher learning rates. Batch normalization also adjusts the model and decreases the demand for Dropout. Lastly, Batch Normalization facilitates the utilization of nonlinearities by avoiding the network from hanging in the saturated modes.

With Batch Normalization, which acts as a regularizer, learning rates become higher, for example reduce the need for Dropout. Batch Normalization utilized in state-of-the-art image classification model lets reach the same accuracy with less training steps. Batch-normalized networks improve the best published result on ImageNet classification (attaining 4.9 % top-5 validation error). Very high learning rate In traditional deep networks can arise in the gradients that erupt or disappear. Batch Normalization aims to solve these problems. Normalizing activations over the network let avoid small fluctuations of the parameters that might increase into larger changes in activations in gradients. With the help of normalizing activation the training does not hang in the nonlinear saturated modes.

Using Batch Normalization training becomes more robust with respect to the parameter scale. Typically, large learning rates of the scale of layer parameters increase and these improves the gradient during backpropagation. It is important to clarify, that backpropagation with Batch Normalization by a layer not distorted by the scale of its parameters, for a scalar *BN (W u) = BN((aW)u)*.

The scale does not influence the layer Jacobian and the gradient propagation. Furthermore, bigger weights lead to smaller gradients and Batch Normalization stabilizes the increasing of parameters. The Batch Normalization can lead the layer Jacobians to have values close to 1, which is advantageous for

training. Corresponding two successive layers with normalized inputs, and the changing between these vectors: $bz = F(bx)$. bx and bz are Gaussian and uncorrelated, $F(bx) \approx Jbx$ is a linear transformation for the given model parameters, therefore bx and bz have unit covariances, and $I = Cov[bz] = JCov[bx]\ JT = JJT$. Consequently, $JJT = I$, and all values of J are equivalent to 1 as well and obtains the gradient sizes throughout backpropagation. In practice, the transformation is not linear and the normalized values are not required (do not have to be) to be Gaussian, but what really surprising is that Batch Normalization can help to make gradient propagation well-behaved [16].

VII. Generalization

Usually the input data for CNN is nonlinear and multidimensional. It raises the question how to properly utilize this data. For this reason in order to gather common features from high-dimensional spaces properly the generalization methods are used. Although regularization together with data augmentation are effective methods to improve the generalization ability of deep CNN, which aim to train the complex models keeping overfitting on low level and extend the datasets in different ways such as translation transformation, horizontal flip, this problem still remains extremely challenging [19].

This section presents considerable progress in understanding deep convolutional neural networks by analyzing of generalization performance and accomplishing optimization for gradient descent based training algorithms [20]. Optimization of deep CNN learning is a serious issue, because of different gradient descent methods and the network structure, including activation functions. Then the actual approach depends on the generalization ability. The development of applications based on deep learning is indivisible from the data support, in particular in image classification, object detection. Human can distinguish inaccuracies in real and natural sample distribution by increasing the amount of training data. But deep CNN overfitting problem still present. The data-driven training method helps us to get the result of data distribution in high-dimensional space during the training process. This mean, that the required reduction of data is only possible with our perception and the efficient extension of the data is a more appropriate method to approximate the natural distribution.It is possible to regularize the feature boundaries of deep CNN in a two-stage training process and therefore to improve the generalization ability of the network [21].

A. Generalization explained by High frequency components

This section describes also the correlation between the generalization behavior of convolutional neural networks and the frequency spectrum of image data. First of all we want to consider CNN's capability in capturing the high-frequency components of images. To understand the generalization behaviors of neural networks it is important to know the properties of stochastic gradient descent, different complexity measures and different models.

We consider the generalization behaviors of CNN from a data perspective. The unsystematic generalization behaviors of

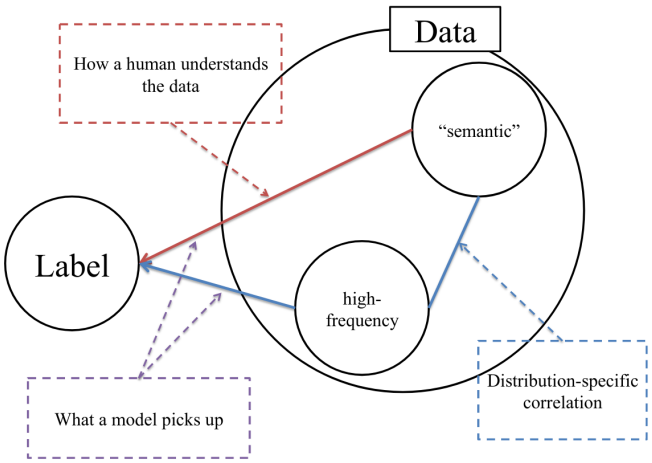

Fig. 7. The central point of Generalization by High frequency components is connections between the high-frequency and the semantic components of the images. As a result, the model considers both high-frequency components as well as semantic, therefore guiding to generalization behaviors that are nontrivial to understand from human pespective. CNN can utilize the high-frequency image components that are not visible to people [22]

CNN is a direct result of the differences in perception between human and models (see Figure 6). CNN can perceive data with a higher granularity than people can do (it could be done by people) [22].

All notations used in this section mean the following: *(x, y)* - a data sample (the image). $f\ (\cdot;\Theta)$ signifies convolutional neural network and the parameters are denoted as Θ. H is stands for a human model, and as an outcome,$f\ (\cdot;\ H)$ describes human classification of the data. $\iota(\cdot,\cdot)$ denotes a generic loss function. $\alpha(\cdot,\cdot)$ is a function evaluating, that forecast accuracy for every sample. d $(\cdot,\cdot)$ is a function, analysing the distance between two vectors.*F (·)* is the Fourier transform and $\mathrm{F}{-}^{1}(\cdot)$ is the inverse Fourier transform. z is used to define the frequency component of a sample. Consequently, $z = F(x)$ and $x = F{-}^{1}(z)$. Just to clarify, Fourier transform can represent complex numbers.

The raw data x = $\{xl, xh\}$, where xl and xh denote the low-frequency component (LFC) and high-frequency component (HFC) of x. Here are the corresponding equations:

$$z = F(x), \qquad Z\iota, Zh = t(z; r), \tag{11}$$

$$X\iota = F{-}^{1}(Z\iota), \qquad Xh = F{-}^{1}(Zh). \tag{12}$$

where *t(·; r)* is a threshold function with the role to divide the low and high frequency components from z correspondingly to a radius *r*. With the help of a one channel image of size $n \times n$ is t(·; r) to determine, with N possible pixel values (x $\in$ $Nn{\times}$n), therefore $z \in Cn \times n$ and C defines the complex number. *z(i, j)* is utilized to index the value of z at position *(i, j)*, and to denote the centroid is used

ci, cj. The following equation *zl , zh = t(z; r)* is determined:

$$Zl(i,j) = \int_0^{z(i,j)} \text{if } d((i,j),(ci,cj)) \leq r \tag{13}$$

otherwise

$$Zh(i,j) = \int_0^{z(i,j)} \text{if } d((i,j),(ci,cj)) \leq r \tag{14}$$

otherwise

$d(\cdot,\cdot)$ in $t(\cdot; r)$ is defined as the Euclidean distance. The approach employs independently on every channel of pixels, in case x has more than one channel. According to assumption that only *xl* is visible to human, but both *xl* and *xh* are visible to a CNN, we determine:

$$y := f\ (x;H) = f(xl;H), \tag{15}$$

but when a CNN is trained with

$$\arg\ \min\iota(f\ (x;\theta),y), \tag{16}$$

which equivalent to

$$\arg\ \min\iota(f\ (\{xl,xh\};\theta),y) \tag{17}$$

To minimize the loss, CNN's must learn to utilize *xh*. As an effect, CNN's generalization behavior compared to a human is unsystematic. It is very important that CNN learns to utilize*xh* which must differ from CNN overfit. The reason is that *xh* can include more data than sample-specific feature and these data can be generalized with the training, but not visible to people [22].

VIII. Applications of CNNs

Deep convolutional neural networks are widely used in many different fields, such as robotics, speech recognition, sentence classification, medicines, economics and others. CNNs using also in robotics with limited computational capabilities. Categorization and object detection is the essential application fields of deep learning in robotics.

State-of-the-art approach utilized for categorization and object detection which based on generating object methods and by classification using a Convolutional Neural Network (CNN). CNN helps systems to recognize various object categories. Real-time operation is important approach for the application of CNNs for characterization and object detection. But object detection and categorization based on CNN methods are not able to operate in real-time in most robotics systems. These approaches are based on using sources for the objects segmentation (depth data, color, etc.) and also utilizing object specific weak detectors for expanding the required proposals. Lightweight and fast CNN architectures can also utilized in case of dealing with a limited number of object categories.

There are two different CNN based NAO robotics detectors that are able to run in real-time. Nao is an autonomous humanoid robot, developed by Aldebaran Robotics. These detectors can analyze a robot object-proposal in 1 ms and the average number of proposals to analyze in the system is 1.5 per frame. The detection rate is approximately 97 %. Deep learning in robotics with limited computational resources must utilize fast and lightweight neural models.

State-of-the-art systems based on CNNs require large memory and computational resources, for example high-end GPUs. CNN-based approaches are incapable to operate on systems with low resources, such as mobile robots. Therefore, it is very important in to research and development the approaches that help CNNs to work with less memory and fewer computational resources, such as quantization of the networks. Different approaches have been offered for the quantization of CNNs. These techniques able to calculate the required convolutions using fast Fourier Transform method. This method utilize representation of the convolutions and compress the parameters of the network. Two binary-based network architectures: XNOR-Networks and Binary-Weight-Networks have shown very gut results. The filters in Binary-Weight-Networks are approximated with binary values in closed form, therefore saving in a 32x memory. The convolutional layers input both filters in XNOR-Networks are binary. However, non-binary non-linearities such as ReLU can be utilized.This outcomes in 58x faster convolutional operations on a CPU, in case of utilizing of XNOR and bit-counting operations. The classification accuracy with a Binary-Weight-Network version of AlexNet is only 2.9% less than the full-precision AlexNet. XNOR-Networks have larger decreasing in accuracy about 12.4%. The best results best results can be achieved by using networks with a low number of parameters in a non-standard CNN structure, like SqueezeNet. Therefore XNOR-Net and SqueezeNet is used for implementing NAO robot detectors [23].

Deep learning models have achieved successful results in speech recognition. For example, in natural language processing, with approach of learning word vector representations by neural language models and learning wordvectors for classification.Convolutional neural networks (CNN) use layers with convolving filters that are utilized to local features. CNN models have been shown as efficient for NLP and have achieved remarkable results in semantic parsing, sentence modeling and other conventional NLP tasks. CNNs were trained with one layer of convolution on top of word vectors received from an unsupervised neural language model. These vectors were trained on 100 billion words of Google News. This model achieves gut results on multiple benchmarks. The simple modification are required for better results and for utilizing of both pre-trained and task-specific vectors.Learning task-specific vectors through fine-tuning results in further improvements [24].

Also Classification of crystallization results using deep convolutional neural networks.The Machine Recognition of Crystallization Outcomes (MARCO) has gathered round half a million images of macromolecular crystallization experiments from different sources. The state-of-the-art approach is utilized here, therefore about 94% of the test images can be properly labeled. Crystal recognition is very important for the systematic analysis of crystallization experiments. This

approach plays an essential role for industrial and fundamental research applications.This model is implemented in TensorFlow and trained using an asynchronous distributed training setup across 50 NVidia K80 GPUs. This approach can generate 100M images (260 epochs) in circa 19 hours. The original labeling increase to a model with 94.2% accuracy . Relabeling enhanced reported classification accuracy by 0.3%. The model reaches 94.5% accuracy [25].

Convolutional Neural Networks is also utilized in economics areas. A stock prediction model and candle charts with continuous time stock data based on deep Convolutional Neural Networks. The original data by prediction time interval and classification approach is divided into various categories as the training set of CNN. The CNN is applied with the goal to predict the stock market and analyse the difference in accuracy under various classification methods. The effects demonstrate that the approach has the best performance by the prediction time interval of 20 days.CNN model needs to be trained, in order to make the forecast outcomes more accurate. CNN is very efficient in addressing the issues of image processing. CNN-based network model have been build for the images of the obtained financial tinformation. This network model consists of four 2d convolutional layers, four 2d pooling layers, and three output layers. CNN requires that the value of a pixel is related to its adjacenting pixels, therefore for CNN is more comfortable to extract texture data and improving the forecast result.The forecast accuracy based on the contained CNN model must be firts compute and therefore analyze the effect of the three indicators of forecast such as interval, classification on the forecast accuracy and image richness. The consequences of that a accurate data set type can be archieved for further investigation in the future [26].

IX. Conclusion

The basis of many modern computer vision models are Convolutional Neural Networks. CNN has made significant progress, particuly iarln image recognition and vision-related tasks. This paper shows similarities and differences in the CNN architectures, the set up of different various layers. CNNs are one of the best algorithms for image recognition and have shown extraordinary performance in image understanding, detection, classification and other related tasks. This paper also covers the architecture classification, training, generalization, its applications and future directions.

The learning performance of CNN is remarkable improved in the last years by utilizing depth and other structural modifications. The latest literature indicate that the important breakthrough in CNN performance has been reached with blocks. One of the main accomplishments of research in CNN architectures is the development of efficient block architectures. The block is very useful in a network. It utilizes three-dimensional or feature-map data and aids for input channels to improve achievement. The blocks play an important role in amplification CNN performance by problem-aware learning.

CNN is able without complicated processing to learn good internal representation from raw pixels. Nowadays most of the developers of image processing and computer vision competitions are applying deep CNN based models. Therefore, deep CNNs have demonstrated considerable performance improvement in recognition tasks which including hundreds of image categories, over conventional vision-based models.

The applying of various progressive ideas in CNN architecture has changed the tendency of research, particularly in image processing. A grid-like topological information changes performance of CNN to the powerful model for images. Deep architectures commonly have a benefit over shallow architectures by complex learning issues. The stacking of multiple linear and non-linear processing units in a layer-wise mode aids to learn complex patterns at various levels of abstraction.

Architecture of CNN is a promising research area. CNN tends to be in the future one of the most broadly used artificial intelligence approaches. The techniques such as batch normalization, dropout are also important for performance of CNNs. The various architectures can help the model in improving generalization and stability on different categories of images by semantic representations. In the future, it is anticipated that the capability of cloud-based platforms will be utilized for the development of CNN applications. In such companies as Google, Microsoft, Facebook and NEC the research groups work active for developing new architectures of CNN [27].

References

[1] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[2] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[4] W. Zhang *et al.*, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of annual conference of the Japan Society of Applied Physics*, 1988.

[5] G. Templeton, "Artificial neural networks are changing the world. what are they?" 2015, [accessed 2020-08-02]. [Online]. Available: https://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world-what-are-they

[6] R. Rojas, "The backpropagation algorithm," in *Neural networks*. Springer, 1996, pp. 149–182.

[7] S. Saha. (2018) A comprehensive guide to convolutional neural networks — the eli5 way. [accessed 2020-09-12]. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.

[9] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017.

[10] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[11] L. Boltzmann, "Studies on the balance of living force between moving material points," *Wiener Berichte. 58*, p. 517–560, 1868.

[12] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*. Springer, 1990, pp. 227–236.

[13] R. Draelos, "Multi-label vs. multi-class classification: Sigmoid vs. softmax," 2019, accessed 2020-09-12. [Online]. Available: https://glassboxmedicine.com/2019/05/26/classification-sigmoid-vs-softmax/

[14] "Different kinds of convolutional filters," 2017, accessed 2020-09-12. [Online]. Available: https://www.saama.com/different-kinds-convolutional-filters/
[15] A.-D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee, "Distribution padding in convolutional neural networks," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4275–4279.
[16] C. S. Sergey Ioffe, "Batch normalization: Accelerating deep network training by reducing internal covariate shift." 2015.
[17] J. S. Rupesh Kumar Srivastava, Klaus Greff, "Training very deep networks." 2015.
[18] A. Rosebrock. (2016) Gradient descent with python. [accessed 2020-09-02]. [Online]. Available: https://www.pyimagesearch.com/2016/10/10/gradient-descent-with-python/
[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[20] J. F. Pan Zhou, "Understanding generalization and optimization performance of deep cnns." 2018.
[21] Q. Zheng, M. Yang, J. Yang, Q. Zhang, and X. Zhang, "Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process," *IEEE Access*, vol. 6, pp. 15 844–15 869, 2018.
[22] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High frequency component helps explain the generalization of convolutional neural networks," 2020.
[23] N. Cruz, K. Lobos-Tsunekawa, and J. Ruiz-del Solar, "Using convolutional neural networks in robots with limited computational resources: Detecting nao robots while playing soccer," *arXiv preprint arXiv:1706.06702*, 2017.
[24] Y. Kim, "Convolutional neural networks for sentence classification," 2014.
[25] A. E. Bruno, P. Charbonneau, J. Newman, E. H. Snell, D. R. So, V. Vanhoucke, C. J. Watkins, S. Williams, and J. Wilson, "Classification of crystallization outcomes using deep convolutional neural networks," *PLOS ONE*, vol. 13, no. 6, p. e0198883, Jun 2018. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0198883
[26] Q. Zhou and N. Liu, "A stock prediction model based on dcnn," *arXiv preprint arXiv:2009.03239*, 2020.
[27] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, pp. 1–62, 2020.