

**CSC 450 – Honors Research Project**

**Student: Dennis Krupitsky**

**Mentor: Dr. Natacha Gueorguieva**

College of Staten Island

[dennis.krupitsky@gmail.com](mailto:dennis.krupitsky@gmail.com)

**Deep Learning Image Recognition and Detection: Architectures, Learning and Applications**

**Deep Learning Image Recognition and Detection: Architectures, Learning and Applications**

**Abstract**

Development of machine learning algorithms allows for solving classic problems such as image classification, detection, and recognition. Utilizing the ability of Deep Learning algorithms, the construction of Deep Neural Networks is possible, which will allow learning of powerful features from huge amounts of data by extracting features of the data layer by layer. The goal of this research is to propose and develop different solutions using Convolutional Neural Networks, by experimenting with different training approaches including batch training, gradient and stochastic gradient descent methods and different activation and loss functions, augmentation, pooling and dropout. Experiments done within this paper use the Flowers data set from Kaggle, which are plotted and analyzed in order to see evaluate the performance of the different applications and architectures using validation procedures.

*Keywords— Machine learning, neural network, deep learning neural networks, CNN, gradients, optimizers, activation functions, K-fold cross validation, supervised learning*

**Table of Contents:**

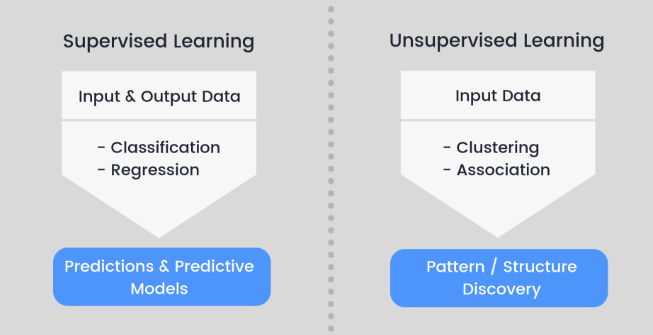
1. Introduction to Deep Learning pg. 3
   1. – Purpose and use of Deep Learning pg. 3
   2. – Neural Networks pg. 4
   3. – Data Preprocessing pg. 5
2. Convolutional Neural Nets
   1. – Activation Functions
   2. – Optimizers
   3. – Confusion Matrix, K-Fold validation
   4. – Mini-batches of data
   5. – Imbalanced Data Sets
   6. – Experiments for overfitting
3. Experimentation
   1. – Dataset information
   2. – Experiment Details
   3. – Experiment Results

**1. Introduction to Deep Learning:**

With machine learning being an ever-growing, popular field within the artificial intelligence world, a subset of machine learning is **Deep Learning.** This specific subset is a learning technique for computers involving algorithms, and neural networks inspired by the human brain to learn from huge amounts of data. The process of creating a model could grow quite extensive as there are many facets to account for. A simple definition for a deep learning model can be described as an algorithm repeatedly performing a task, and each time have certain tweaks in order to improve the outcome. The “*deep*” within deep learning is a reference to the number of successive layers of representations, which could also he described as the *depth* of the model. A model could range from one to hundreds of successive layers, all learned during the exposure to training data. Deep learning has only come to surface as one of the most useful AI techniques in the last few decades, as we now have access to large amounts of labeled data for training (over 2 quintillion bytes of data is generated daily), and substantial computing power to train our models. Overall, deep learning allows modern machines to solve complex problems, by learning from experience.

**1.1: Purpose and use of Deep Learning:**

Deep learning is one of the areas that has attracted a lot of attention due to its potential for real world applications. It is widely used in real life applications, such as image classification, aerospace and defense, medical research, self-driving cars, robots, etc. There are several forms in which this type of machine learning can be trained. One allows for a training method with data that is pre-labeled, and through training the model is comparing the label it assigned the data to the actual label, to see if its prediction was correct or not, also known as supervised learning. As mentioned earlier, there is tons of data that is collected every day, but most of this data is not labeled, so we are not able to use it in supervised learning. This is where unsupervised learning comes in to play, we are still able to show the data to our deep learning networks, and it will learn to identify the data’s label.

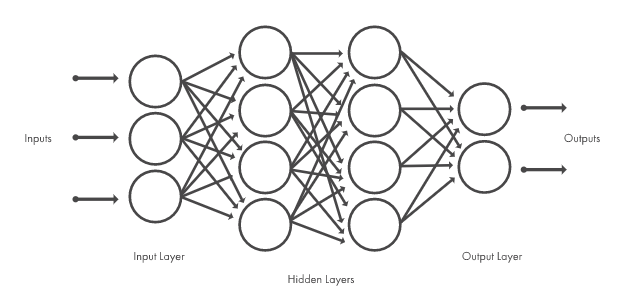


**Fig 1. Supervised vs Unsupervised learning**

These networks can be successfully applied to huge amounts of data for knowledge discovery, application of this knowledge, predictions based off the knowledge, etc. Deep learning is used to create actionable results. Deep learning allows us to advance and innovate within the real world, and is one of the most powerful aspects of machine learning.

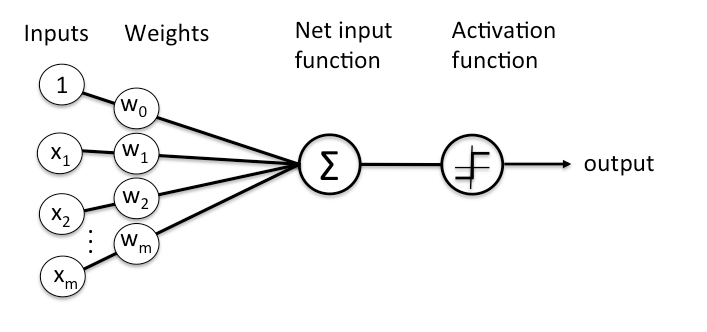
**1.2: Neural Networks:**

Deep learning got its name for another reason, more specifically due to the neural networks it is comprised of have various deep layers that enable learning. A neural network is a set of algorithms, which as stated earlier are based off the human brain, which we design in order to recognize patterns, whether it be in images, text, etc. Neural networks assist us in clustering and classifying data. These neural networks are a set of layers that are stacked on top one-another that adjust to the properties of the training data.



**Fig 2. Representation of a Neural Network**

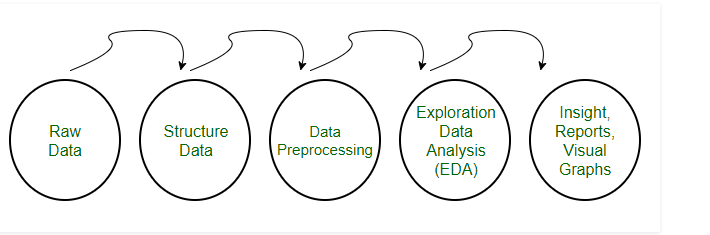
The input layer is the initial data being brought in that will be used by the neural network. The hidden layers within neural networks make this one of the superior machine learning algorithms. These hidden layers are not visible to external systems, and are private to the neural network. The amount of these hidden layers can also range from zero to hundreds. Each hidden layer is comprised of neurons that receive an input from the previous layer, and does some sort of manipulation or transformation to the data before sending it to the next layer/neuron. The final layer in the flow chart is the output layer, which produces the result for the original inputs. Essentially each layer, start at the hierarchy, is combining information into something more and more complex, depending on the number of layers you utilize. Each node in a layer relates to each node in the following layer, and each arrow in the connection holds a certain weight. This could also be perceived as the impact that the node has on the next layers node.



**Fig. 3 Representation of a single node/layer**

**1.3 Data Preprocessing:**

There is a crucial step that is taken before the training of a machine learning model, which helps improve both the quality of our data, and result of the model. This step is called data preprocessing, a data mining technique used to transform raw data into usable formats. Within it there are several steps, such as data cleansing, data transformation, data distribution, etc. These steps allow our data to take a form in which they can create a model. We use this in order to combat data inefficiencies which could have negative effects on our experiments, such as inaccurate data, noisy data, inconsistent data, etc. If this step is skipped over, there is a possibility that a percentage of the results will be false.



**Figure 4: Data Preparation**

Some of the widely used techniques include removing null data, rescaling data, standardizing data, binarizing data, and label encoding. Missing or null values should be handled properly during data preprocessing, in order to avoid altered results that will differ from the proper data. Rescaling data is the process of converting data that is comprised of attributes with varying scales, into common values which range between 0 and 1. This method is very useful in optimization algorithms. Standardizing data allows the transformation of values with a Gaussian distribution of differing means and of differing standard deviations into a standard Gaussian distribution that has a mean of 0 and standard deviation of 1. The formula looks as following:

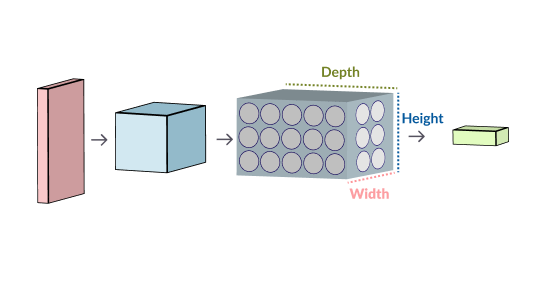


**Figure 5: Standardization Formula**

Binarizing the data allows all values over the threshold to be marked 1, while all equal to or below the threshold are marked as 0. This method is useful when dealing with probabilities, as it allows the data to be transformed into crisp data. Label encoding transforms data labels, which are usually labeled with words in order to make it readable, into numbers or binary labels for the algorithms to be able to work with them. Another crucial step is to distribute the original collection of raw data into separate sets, more specifically training, validation, and testing sets. The training dataset contains data that will be used to train the model, as the model sees and learns from this data, therefore this set should have the biggest ratio of data. The validation dataset is used to evaluate a given model during its training, this data is used to fine-tune the hyperparameters, this set should have slightly more data than the testing from the remaining available data after the training set is allocated. The testing set is used to fully evaluate the model, after it has completed training (using test and validation sets). It should receive the remaining undistributed data. It is good practice to not have 2 sets containing the same data. Once the data has been preprocessed visual graphs, and reports are generated for the researcher to get a sense of the altered data’s values

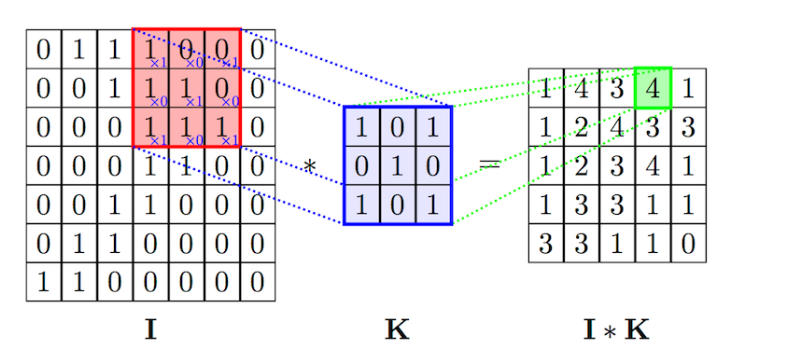
**2. Convolutional Neural Network:**

One of the main components of this image processing research experiment involves convolutional neural networks. A convolutional neural network (CNN) is a deep learning algorithm which is comprised of neurons that can take an image as input, assign certain weights, and biases to the aspects/objects within the image, and successfully differentiate from one another. Like a standard neural network, a CNN also consists of an input layer, an output layer, additionally adding on pooling layers, convolutional layers, normalization layers, fully connected layers, etc. CNN architectures make the explicit assumption that the input it is going to be receiving will be images, which then allow for it to encode certain properties into the architecture. Allowing for the architecture to be focused to a certain type of data allows for an increase in efficiency for image processing results. There are a few differences between CNNs and regular neural networks. As mentioned earlier a neural network will transform an input by sending it through a series of hidden layers within the model. These hidden layers are of course comprised of sets of neurons, where the layers are connected to the layer preceding it. A main difference in Convolutional Neural Networks is that the layers within it are organized into 3 different dimensions of width, height, and depth. Another difference is that the neurons within a layer do not all connect to neurons in the next layer, but instead only to a small region of it. Instead within this 3-dimensional structure, each set of neurons analyzes is set to analyze a specific region of the image. Within the figure pictured below the red input layer represents the image, with the width and height being the dimensions, and the depth having a value of 3 (Red, green, blue) channels.



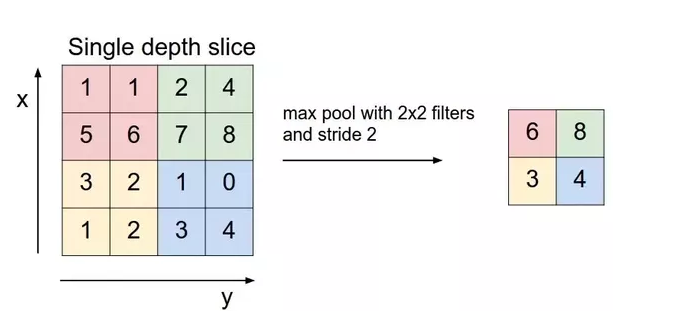
**Figure 6: Convolutional Neural Network Structure**

The first layer following the input layer is the convolutional layer that allows a feature map to be produced. Convolution is a mathematical operation on two function that will produce a third function. The purpose of this layer is to be able to figure out certain features that an image contains, for instance, the vertical/horizontal edges, gradients, etc. Input to this layer is the (m x m x r) image, where m represents the height and the width, and r is the depth or number of channels, usually this input will be as an array of pixel values. This layer will also define a filter/kernel of size (n x n x q), where n is defined to be smaller than the dimension of the image, and q is defined to have the same channels r. Essentially this convolutional layer allows the filter to slide across the input, and at every location a matrix multiplication will occur and then sums the result onto the feature map with the process then repeating for every location on the input volume. Within a CNN, there could be several Convolutional layers of varying kernel sizes.



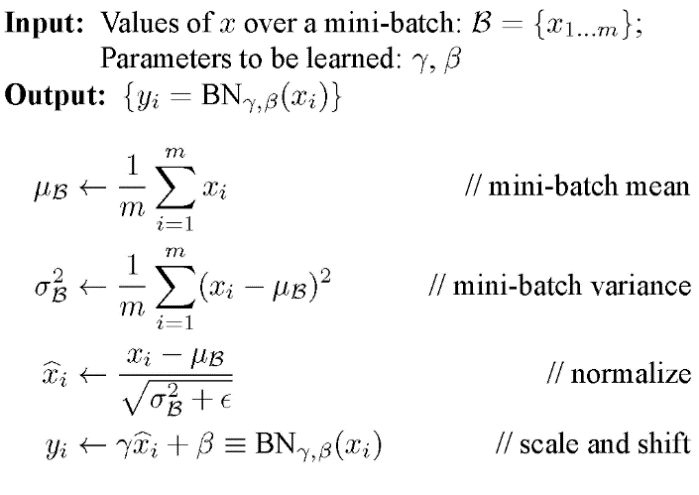
**Figure 7: Feature Map**

Following a convolutional layer it is usually common to add a pooling layer in between CNN layers. It serves the function of reducing the size of the matrix in order to reduce the amount of spatial size produced from the convolved feature map. This allows for a decrease in the number of parameters and computation power required to process the data. The most common used form of pooling is max-pooling. Max pooling takes the maximum value amongst each kernel of the feature map, which in turn allows the feature map to decrease in size, but also retain the significant information. Max pooling also serves as a noise suppressant, because it discards the noisy activations while also performing dimensionality reduction. Similar to convolutional layers, you can include several of these pooling layers within your network, and as a result there will be deeper extraction of features within the images.



**Figure 8: Max Pooling**

Another layer that is often included within CNNs, and used within this experiment is a batch normalization layer. This normalizes each of the input channels across a mini batch by adjusting and scaling the activations. Additionally, batch normalization allows for each individual layer of the network to learn by itself a bit more independently from other layers. During training of the network, any activation and distribution changes within a layer due to the changing weights and biases will cause rapid changes in the layer above it, and cause training to slow down. As an example, when there are features that could range from 0-1, and then features that could range from 1-1000. The normalization of these values will allow of an increase in speed during training. Along with speed it also reduces the sensitivity of network initialization when training convolutional neural networks. Batch normalization is usually placed between each convolution layer within the network. Including batch normilization is always good, as it serves as almost a preprocessing step at every layer within the network. The formula to calculate the normalization values goes as following:



**Figure 9: Batch Normilization formula**

**2.1: Optimizers:**