CONTENTS

LIST OF ABBREVIATIONS

NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The field of machine learning has taken a dramatic twist in recent times, with the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the performance of previous forms of artificial intelligence in common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offers a simplified method of getting started with ANNs.

Convolutional neural networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition, dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

When applying CNNs to a computer vision task, a change in viewpoint (change in orientation, position, shear, etc.) is likely to lead to drastically different network activations, hindering the model's ability to generalize. To solve this problem, current CNNs require a large number of parameters, datasets and computational power. This lead to the introduction of Capsule Networks. Capsule Networks aim to generalize to different viewpoints by taking advantage of the fact that the relationship between parts of an object is viewpoint invariant. It has been shown that these networks generalize better than standard CNNs, are more robust to adversarial attacks, achieve higher accuracy, all while requiring significantly fewer parameters.

CHAPTER 2

ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks (ANN) are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

ANN works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

Neural networks help us cluster and classify. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification.

Above all, these neural nets are capable of discovering latent structures within unlabeled, unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing and clustering the world's raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to.

2.1 Comparison of Neural Networks with Conventional Computing

A serial computer has a central processor that can address an array of memory locations where data and instructions are stored. Computations are made by the processor reading an instruction as well as any data the instruction requires from memory addresses, the instruction is then executed and the results are saved in a specified memory location as required. In a serial system (and a standard parallel one as well) the computational steps are deterministic, sequential and logical, and the state of a given variable can be tracked from one operation to another.

In comparison, ANNs are not sequential or necessarily deterministic. There are no complex central processors, rather there are many simple ones which generally do nothing more than take the weighted sum of their inputs from other processors. ANNs do not execute programed instructions; they respond in parallel (either simulated or actual) to the pattern of inputs presented to it. There are also no separate memory addresses for storing data. Instead, information is contained in the overall activation 'state' of the network. 'Knowledge' is thus represented by the network itself, which is quite literally more than the sum of its individual components.

2.2 Working of ANN

Artificial Neural Networks can be viewed as weighted directed graphs in which artificial neurons are nodes, and directed edges with weights are connections between neuron outputs and neuron inputs.

The ANN receives information from the external world in the form of patterns and images in vector form. These inputs are mathematically designated by the notation x(n) for n number of inputs. Each input is multiplied by it's corresponding weights. Weights are the information used by the neural network to solve a problem. Typically weight represents the strength of the interconnections between neurons inside the neural network. The weighted inputs are all summed up inside computing unit (artificial neuron). In case the weighted sum is zero, bias is added to make the output non-zeroor to scale up the system response. Bias has the weight and input always equal to '1'. The sum corresponds to any numerical value ranging from 0 to infinity. To limit the response to arrive at the desired value, the threshold value is set up. For this, the sum is passed through an activation function. The activation function is set to the transfer function used to get the desired output. There are linear as well as non linear activation functions. Some of the commonly used activation functions are -

- Sigmoid or Logistic Activation Function - The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.
- Tanh or hyperbolic tangent Activation Function - tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1).
- ReLU (Rectified Linear Unit) Activation Function - The ReLU is the most used activation function since it is used in almost all the convolutional neural networks or deep learning. The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. Range: [0 to infinity)

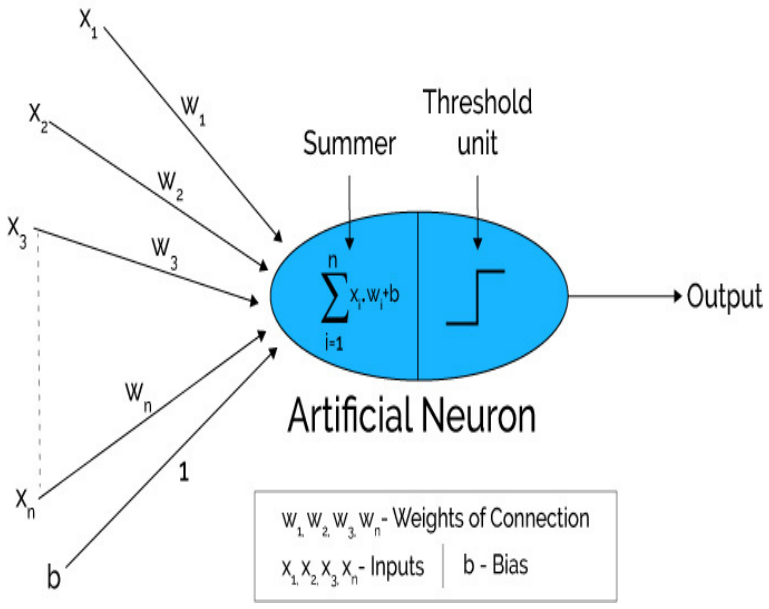
X_1
X_2
X_3
X_n
b
W_1
W_2
W_3
W_n
1
Summer
Threshold unit
$\sum_{i=1}^{n} x_i.w_i+b$
Output
Artificial Neuron
$w_1 w_2 w_3 w_n$ - Weights of Connection
$x_1 x_2 x_3 x_n$ - Inputs
b - Bias

Figure 2.1: Working of Artificial Neural Network

2.3 Neural Network Architecture

A typical Neural Network contains a large number of artificial neurons called units arranged in a series of layers.

A typical ANN comprises of different layers as shown in the figure -

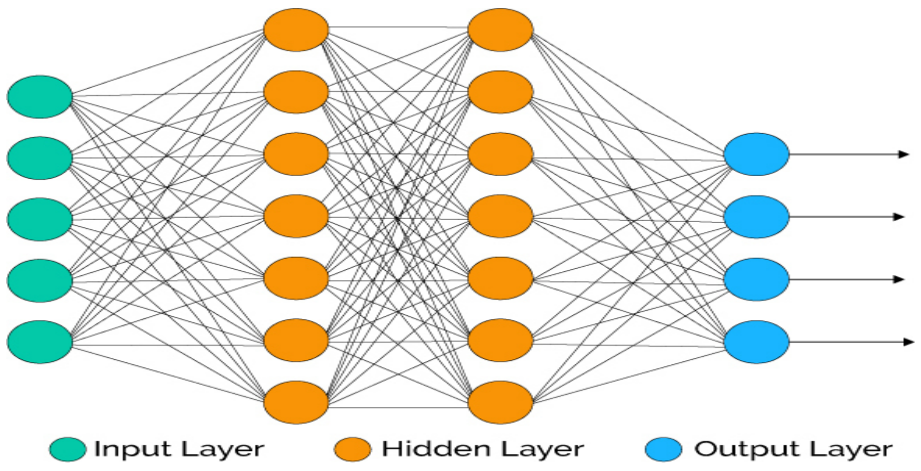

Figure 2.2: Neural Network Architecture

- Input layer – It contains those units (Artificial Neurons) which receive input from the outside world on which network will learn, recognize about or otherwise process.
- Output layer – It contains units that respond to the information about how it's learned any task.
- Hidden layer – These units are in between input and output layers. The job of the hidden layer is to transform the input into something that output unit can use in some way.

Most Neural Networks are fully connected that means to say each hidden neuron is fully linked to every neuron in its previous layer(input) and to the next layer (output) layer.

CHAPTER 3

CONVOLUTIONAL NEURAL NETWORK

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image and Video recognition, Image Analysis and Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

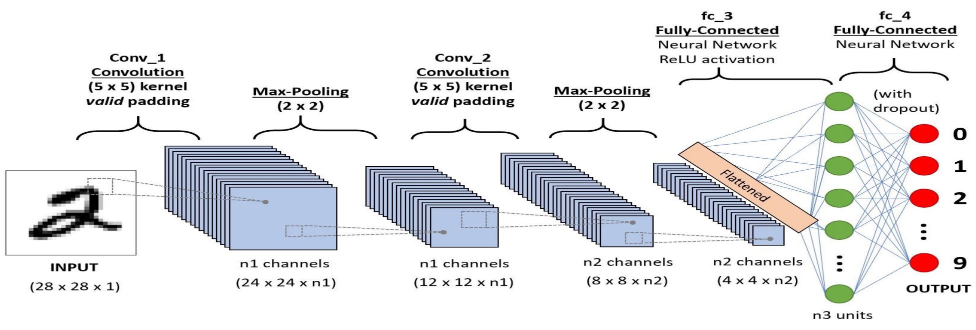

Figure 3.1: CNN architecture for digit classification

3.1 Input Image

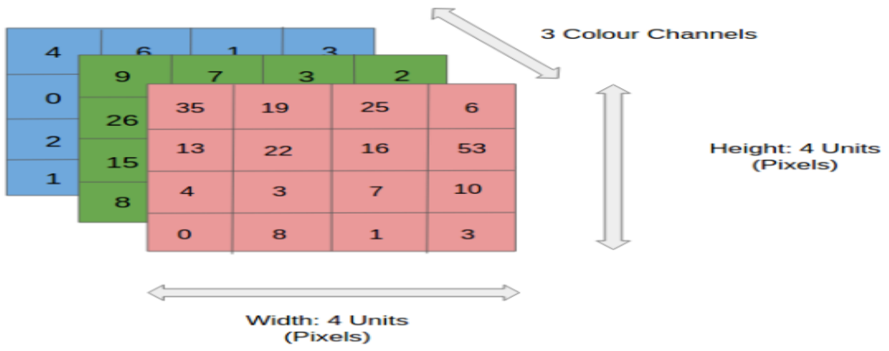

Figure 3.2: Sample input image

In the figure 3.1, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc. CNN image classifications takes an input image, processes it and classify it under certain categories. Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

3.2 Convolutional Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

3.3 Stride

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on.

3.4 Padding

Sometimes filter does not fit perfectly fit the input image. There are two options -

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit.

3.5 Non linearity

ReLU stands for Rectified Linear Unit for a non linear operation. The output is f(x) = max(0,x). ReLU's purpose is to introduce non-linearity in our CNN. Since, the real world data would want our CNN to learn would be non-negative linear values.

There are other non linear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than other two.

3.6 Pooling

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types -

- Max Pooling - takes the largest element from the rectified feature map
- Average Pooling - takes the average of all elements from the rectified feature map
- Sum Pooling - takes the sum of all elements in the feature map

3.7 Fully Connected Layer

This layer is called as FC layer, it involves flattening the matrix into vector and feed it into a fully connected layer like neural network.

Feature map matrix will be converted as vector (x1, x2, x3, ...). With the fully connected layers, these features are combined together to create a model. Finally, an activation function such as softmax or sigmoid is used to classify the outputs according to it's labels.

3.8 Drawbacks

It's true that CNNs have shown great success in solving object recognition and classification problems. However, they aren't perfect. There are certain problems that arise from areas that CNN care not able to solve.

The main reason that CNNs did not work well initially was because they could only make predictions on an image if the image that they were trained on and the test image were almost perfectly aligned. What this means is that if a CNN is trained on images of dogs sitting in a certain position in a certain part of an image, then they will only be able to identify dogs in test images that are sitting in similar positions in similar parts of the image. Even a slight position shift will throw off the CNN's prediction. CNNs are very bad at encoding these different representations of pose and orientation within themselves.

A CNN makes predictions by looking at an image and then checking to see if certain components are present in that image or not. If they are, then it classifies that image accordingly. .Imagine a face. What are the components? We have the face oval, two eyes, a nose and a mouth. For a CNN, a mere presence of these objects can be a very strong indicator to consider that there is a face in the image. However, if asked to predict what the image on the right is, it will still identify it as a face! It does this because the CNN only checks to see if certain features, like eyes, ears, a mouth, and a nose, are present in the image. However, the CNN does not check the relative locations of these features to each other. This shortcoming can be traced back to the pooling layer. In the pooling layer, we generally take the input vector and then select the largest pixel intensities in our receptive field. However, at the same time, the pooling layer loses a lot of the positional information about a feature. It does not take into account the relationship of objects with the surroundings and their orientation to identify objects.

Finally, how the CNN routes data from the lower levels to the higher levels are fundamentally wrong. In a CNN, all low-level details are sent to all the higher-level neurons. These neurons then perform further convolutions to check whether certain features are present. Professor Hinton argues that instead of having the information go through all the neurons, like in a conventional CNN, it is better to route the image to specific group of neurons that have the capability to deal with those features.

CHAPTER 4

CAPSULE NETWORK

A Capsule Neural Network (CapsNet) is a machine learning system that is a type of ANN that can be used to better model hierarchical relationships. The approach is an attempt to more closely mimic biological neural organization.

Professor George Hinton, Sarah Sabour and Nicholas Frosst borrowed ideas from neuroscience that suggest that the brain is organized into modules called capsules. These capsules are particularly good at handling features of objects. The brain, they theorize, must have a mechanism for routing low-level visual information to what it believes is the best capsule for handling it.

The idea is to add structures called “capsules” to a CNN, and to reuse output from several of those capsules to form more stable (with respect to various perturbations) representations for higher order capsules. The output is a vector consisting of the probability of an observation, and a pose for that observation. This vector is similar to what is done for example when doing classification with localization in CNNs.

4.1 Capsules

A capsule is a set of neurons that individually activate for various properties of a type of object, such as position, size and hue. Formally, a capsule is a set of neurons that collectively produce an activity vector with one element for each neuron to hold that neuron’s instantiation value (e.g., hue). Graphics programs use instantiation value to draw an object. Capsnets attempt to derive these from their input. The probability of the entity’s presence in a specific input is the vector’s length, while the vector’s orientation quantifies the capsule’s properties.

Artificial neurons traditionally output a scalar, real-valued activation that loosely represents the probability of an observation. Capsnets replace scalar-output feature detectors with vector-output capsules and max-pooling with routing-by-agreement.

Because capsules are independent, when multiple capsules agree, the probability of correct detection is much higher. A minimal cluster of two capsules considering a six-dimensional entity would agree within 10% by chance only once in a million trials. As the number of dimensions increase, the likelihood of a chance agreement across a larger cluster with higher dimensions decreases exponentially. Capsules in higher layers take outputs from capsules at lower layers, and accept those whose outputs cluster. A cluster causes the higher capsule to output a high probability of observation that an entity is present and also output a high-dimensional vector.

4.2 Equivariance

Equivariant properties such as a spatial relationship are captured in a pose, data that describes an object's translation, rotation, scale and reflection. Translation is a change in location in one or more dimensions. Rotation is a change in orientation. Scale is a change in size. Reflection is a mirror image. Unsupervised capsnets learn a global linear manifold between an object and its pose as a matrix of weights. In other words, capsnets can identify an object independent of its pose, rather than having to learn to recognize the object while including its spatial relationships as part of the object. In capsnets, the pose can incorporate properties other than spatial relationships, e.g., color (cats can be of various colors). Multiplying the object by the manifold poses the object (for an object, in space). It learns to detect a particular object (e.g., a rectangle) within a given region of the image, and it outputs a vector (e.g., an 8-dimensional vector) whose length represents the estimated probability that the object is present[1], and whose orientation (e.g., in 8D space) encodes the object's pose parameters (e.g., precise position, rotation, etc.). If the object is changed slightly (e.g., shifted, rotated, resized, etc.) then the capsule will output a vector of the same length, but oriented slightly differently. Thus, capsules are equivariant.

4.3 Routing by agreement

A capsule network is built up by combining several layers. The Capsules present in the lower level of capsule network architecture correspond to simple entities that further combine to form an image. For example, in the image of a house (complex entity), the two smaller entities are rectangle and triangle. According to the concept of Routing by Agreement as specified by Geoffrey Hinton, the capsules at lower level bet on the presence of a complex entity at a higher level. And these bets are combined to get the output of capsules at higher level. For example, the presence of two simple entities, rectangle and triangle, work with each other to bet on the presence of a complex entity at a higher level i.e. house. Here, capsules at lower level try to predict the output of the capsule at the higher level and if the predictions made by capsules at lower level conform to the actual output of capsule at higher level, the coupling coefficient between the two capsules increases.

4.4 How the vector inputs and outputs of a capsule are computed

We want the length of the output vector of a capsule to represent the probability that the entity represented by the capsule is present in the current input. We therefore use a non linear "squashing" function as shown in the Eq.4.1 to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1.

$$\mathbf{v}_j = \frac{||s_j||^2}{1+||s_j||^2}\frac{s_j}{||s_j||} \tag{4.1}$$

where $\mathbf{v}_j$ is the vector input of capsule j and s_j is its total input.

For all but the first layer of capsules, the total input to a capsule s_j is a weighted sum over all “prediction vectors” $\mathbf{u}_{j|i}$ from the capsules in the layer below and is produced by multiplying the output $\mathbf{u}_i$ of a capsule in the layer below by a weight matrix $\mathbf{W}_{ij}$ depicted in Eq.4.2

$$s_j = \sum_{i=1} c_{ij}\mathbf{u}_{j|i}, \qquad \mathbf{u}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \tag{4.2}$$

where the c_{ij} are coupling coefficients that are determined by the iterative dynamic routing process.

The coupling coefficients between capsule i and all the capsules in the layer above sum to 1 and are determined by a “routing softmax” whose initial logits b_{ij} are the log prior probabilities that capsule i should be coupled to capsule j.

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{4.3}$$

The log priors in the Eq.4.3. can be learned discriminatively at the same time as all the other weights. They depend on the location and type of the two capsules but not on the current input image. The initial coupling coefficients are then iteratively refined by measuring the agreement between the current output v_j of each capsule, j, in the layer above and the prediction $\mathbf{u}_{j|i}$ made by capsule i.

The agreement is simply the scalar product $a_{ij} = v_j \ . \ \mathbf{u}_{j|i}$. This agreement is treated as if it was a log likelihood and is added to the initial logit, b_{ij} before computing the new values for all the coupling coefficients linking capsule i to higher level capsules.

In convolutional capsule layers, each capsule outputs a local grid of vectors to each type of capsule in the layer above using different transformation matrices for each member of the grid as well as for each type of capsule.

Algorithm 1: Routing Algorithm

```
Input  : The input vectors to each capsule
Output: The vector outputs of the last capsule layer
1  procedure ROUTING($u_{j|i}$,$r$,$l$) ;
2  for all capsule i in layer l and capsule j in layer l+1: do
3      $b_{i|j} \leftarrow 0$ ;
4  for r iterations do
5      for all capsule i in layer l do
6          $c_i \leftarrow softmax(b_i)$
7      for all capsule j in layer l+1 do
8          $s_j \leftarrow \Sigma_i c_{ij} \mathbf{u}_{j|i}$
9      for all capsule j in layer l+1 do
10         $\mathbf{v}_j \leftarrow squash(s_j)$
11     for all capsule in layer l and capsule j in layer (l+1) do
12         $b_{ij} \leftarrow b_{ij} + \mathbf{u}_{j|i}.\mathbf{v}_j$
13     return $\mathbf{v}_j$
```

4.5 Margin loss for digit existence

We are using the length of the instantiation vector to represent the probability that a capsule's entity exists. We would like the top-level capsule for digit class k to have a long instantiation vector if and only if that digit is present in the image. To allow for multiple digits, we use a separate margin loss, L_k for each digit capsule, k:

$$L_k = T_k max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda(1 - T_k) max(0, \|\mathbf{v}_k\| - m^-)^2 \qquad (4.4)$$

where $T_k =$ iff a digit of class k is present and $m^+ = 0.9$ and $m^- = 0.1$. The λ down-weighting of the loss for absent digit classes stops the initial learning from shrinking the lengths of the activity vectors of all the digit capsules. We use $\lambda = 0.5$. The total loss is simply the sum of the losses of all digit capsules.

4.6 Capsnet Architecture

A simple Capsnet is shown in the Figure 4.1. The architecture is shallow with only two convolutional layers and one fully connected layer. Conv1 has 256, 9x9 convolutional kernels with a stride of 1 and ReLU activation. This layer converts pixel intensities to the activities of local feature detectors that are then used as inputs to the *primary* capsules.

The primary capsules are the lowest level of multi-dimensional entities and, from an inverse graphics perspective, activating the primary capsules corresponds to inverting the rendering process. This is a very different type of computation than piecing instantiated parts together to make familiar wholes, which is what capsules are designed to be good at. The second layer (PrimaryCapsules) is a convolutional capsule layer with 32 channels of convolutional 8D capsules (i.e. each primary capsule contains 8 convolutional units with a 9x9 kernel and a stride of 2). Each primary capsule output sees the outputs of all 256x81 Conv1 units whose receptive fields overlap with the location of the center of the capsule. In total PrimaryCapsules has [32x6x6] capsule outputs (each output is an 8D vector) and each capsule in the [6x6] grid is sharing their weights with each other. One can see PrimaryCapsules as a Convolution layer with Eq.(4.1) as its block non-linearity. The final Layer (DigitCaps) has one 16D capsule per digit class and each of these capsules receives input from all the capsules in the layer below.

We have routing only between two consecutive capsule layers (e.g. PrimaryCapsules and DigitCaps). Since Conv1 output is 1D, there is no orientation in its space to agree on. Therefore, no routing is used between Conv1 and PrimaryCapsules. All the routing logits (b_{ij}) are initialized to zero. Therefore,initially a capsule output (u_i) is sent to all parent capsules (v_0 ...v_9) with equal probability (c_{ij}).

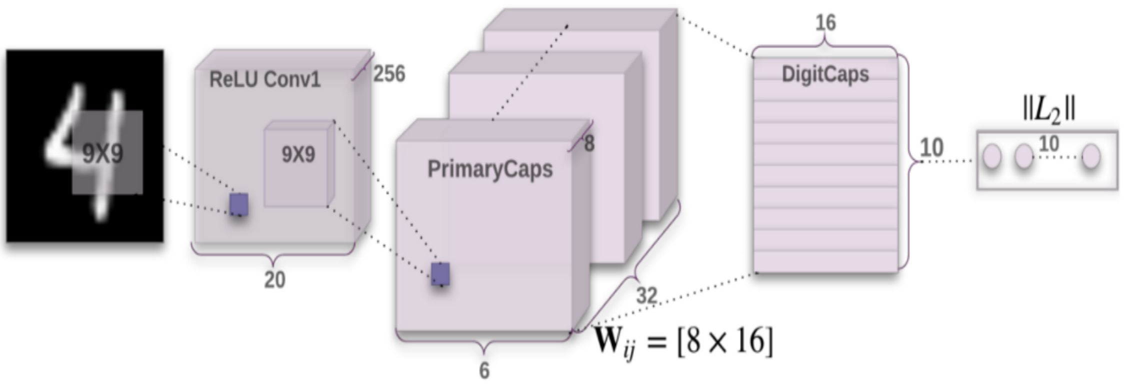

Figure 4.1: A simple CapsNet with 3 layers

This model gives comparable results to deep convolutional networks. The length of the activity vector of each capsule in DigitCaps layer indicates presence of an instance of each class and is used to calculate the classification loss. W_{ij} is a weight matrix between each u_i , $i \in$ (1, 32x6x6) in Primary Capsules and v_j , $j \in$ (1, 10).

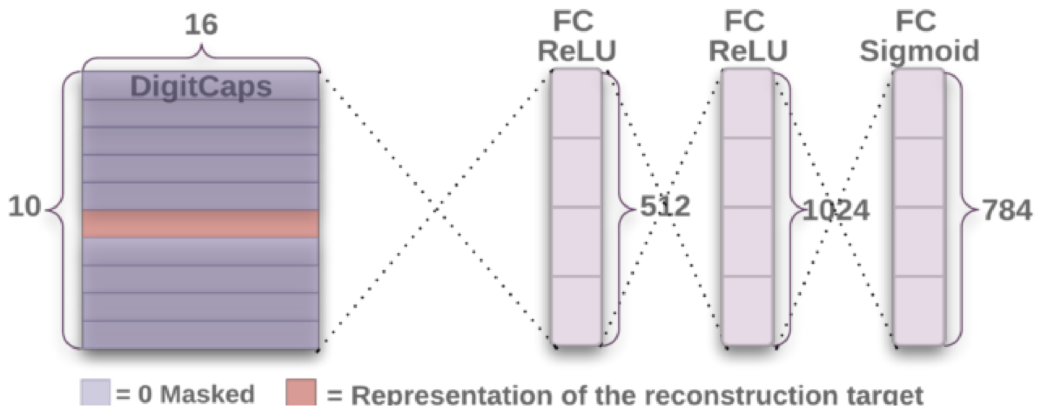

Figure 4.2: Decoder structure

Decoder structure to reconstruct a digit from the DigitCaps layer representation. It is shown in the Figure 4.2. The euclidean distance between the image and the output of the Sigmoid layer is minimized during training. We use the true label as reconstruction target during training.

4.7 Applications

Deep Learning architecture are now being applied in all the domains for improving the accuracy and efficiency of prediction and classification. They are also applied in the field of NLP for Sentiment Analysis and Text Classification. Convolutional Neural Networks are used for learning the numerical representation of words and sentences and they have outperformed the baseline methods such as Bag of words Model, TF-IDF model etc used for the purpose of Text Classification. Similarly, recurrent neural network and its variations, LSTM and auto encoders are used for text classification because of the sequential nature of text.

CNN combined with Recurrent Neural Network also perform text classification tasks. The model described in [5] has a bidirectional recurrent unit at the first level that captures the contextual information, and max pooling layer at the second level captures the features that have best values to perform the classification. Word2Vec skip-gram method was used to train the word-embedding and was given as an input to the model. The model proved to be very effective for computing semantic text representation and can capture more contextual information in comparison to other traditional methods.

Automatic document classification tasks can be done easily using the combination of numerical models, Word2Vec and Machine Learning Algorithms. The combination of these classification models is being applied to Hierarchical Text Classification and their results prove that FastText has been an outstanding method as a classifier, experiments and evaluations were done on the RCV1 dataset. [6, 7]

Text Clustering Algorithm is also proposed by the researchers to classify the textual data without have the labeled datasets using deep learning algorithms. The Algorithm makes use of transfer learning domain adaptation and updating the parameters during the cluster iteration. [8] The model pretraining problem has been resolved using this algorithm in the unsupervised text classification problem. According to their conclusion, the clustering accuracy of their proposed model is superior to other similar algorithms. Research is being done on the application of Capsule Networks in Natural Language Processing for learning the distributed representation of words and sentences. Many research papers have been published related to the application of capsule networks for text classification, sentiment analysis and prediction.

Capsule Networks are also applied in big data analytics. A research paper has been published that discusses about video analytics using Capsule Networks.

In 2017, Geoffrey Hinton, Sara Sabour and Nicholas Frosst, published a paper "Dynamic Routing between Capsules" [1] that discussed about the drawbacks of Convolution Neural Network and introduced a new deep neural network architecture called "Capsule Networks" that overcome the limitations of Convolution Neural Networks. [9] After the publication of this type of neural networks, researchers started working on the application of Capsule networks in different domains. The Capsule Layer replaces the scalar-output feature maps generated by convolutional layer by the vector-output capsules in order to represent the semantics of words and local order of words using instantiated parameters. [5] One such domain is Text Classification and Sentiment Analysis.

In [10] they have applied capsule networks for text classification and have proposed a simple routing method called static routing that has been successful in reducing the computational complexity of dynamic routing and resulted in higher classification accuracies. They have used seven benchmark datasets for the application of capsule networks combined with their proposed routing method and have presented the comparisons and results.

They have another key contribution as well, where they have proposed the use of an ELU-gate for the purpose of propagating relevant information. Different hyper parameters, such as Batch size, Regularization constant, number of filters, filter size, Initial Learning Rate, Number of Capsules and dimension of capsules have been used while performing the experiments. According to their conclusion, capsule networks have outperformed the CNN in text classification achieving 87.17 % accuracy and are indeed useful.

In [2], capsule networks with dynamic routing for the purpose of text classification have been explored. They have proposed three strategies in this paper to reduce the effect of noise capsules. They have developed a model that majorly consists of four layers, primary capsule layer, n-gram convolutional layer, fully connected capsule layer and convolutional capsule layer in a particular order. [2] Six different benchmark datasets for classification have been used for performing the experiments. According to their results, Capsule networks have excelled in four out of six datasets. Their model has achieved 92.6% text classification accuracy on AG's News Dataset performing news categorization. Capsule Networks address the issues observed in convolutional neural network. They have the ability to encode the spatial patterns and also keep the representation capability flexible. They have also shown that capsule networks perform better in multi-label text classification over single-label text classification.

Another research work that has been proposed in this domain is related to identify aggression and toxicity in the comments using capsule networks. In [3] They have proposed a single layer model capsule network along with focal task in order to identify toxicity in the comments. Their model has achieved competitive results in comparison to the other strong baseline methods. They have also shown that the problem occurrence during extensive pre-processing and augmentation of data can be tackled using capsule networks. Their model has four layers, Word Embedding Layer, Feature Extraction Layer, Capsule Layer and The Convolutional Capsule. They have achieved an overall ROC AUC of 98.46, and have shown that their model has beaten other architectures by a good margin on the Kaggle dataset of toxic comments.

A research paper has been published on Sentiment Analysis by Capsules; they have used the combination of Recurrent Neural Network and Capsule Network for sentiment analysis. They have built one capsule for a particular sentiment category, 'positive' and 'negative'. Attention Mechanism is used to build the capsule representation in the implementation of the representation module. They have used two benchmark datasets (Movie Review and Stanford Sentiment Treebank) and another is a proprietary dataset (Hospital Feedback). The structure of capsule used in their model gives it more capacity to model the sentiments. Moreover, their model does not require any kind of linguistic knowledge in order to output the sentiment tendencies. They have designed a capsule that consists of a state, an attribute and three modules. They have shown that their proposed model RNN-Capsule has achieved state-of-the-art performance on classification of sentiment achieving 91.6% accuracy. [4]

Research has also been done in the areas of Image Classification, Computer Vision and data analytics using Capsule Networks. This neural network architecture has shown a good and effective performance in all the domains and has overcome the limitations of traditional neural network architectures. [11, 12]

CHAPTER 5

CONCLUSION

There are different domains in which capsule network is applied and has proved to be efficient in comparison to Convolution Neural Networks. Researchers are still working on this area and publishing papers in the journals. Since, it is a very new topic that has been introduced in 2017, not much work has been done. We believe this new neural network architecture needs to be explored. We have theoretically studied the differences between the Capsule Network and Convolutional Neural Network. We have discussed the points why convolutional neural network lacks behind in certain areas. The scope of Capsule Networks is quite large. These newly developed neural network architectures can be applied in Hierarchical text classification and improve the results in this area. They can also be further applied in detecting emotions from the textual data that is generally termed as Sentiment Analysis in the areas of research and development. Capsule Networks has the capacity to replace the convolutional neural networks in future and would result in greater accuracy and precision in different domains where deep learning models have been applied.

REFERENCES

[1] S. Sabour, C. V Nov, and G. E. Hinton, “Dynamic Routing Between Capsules,”arXiv:1710.09829 Nips, 2017.

[2] Wei Zhao and Jianbo Ye and Min Yang and Zeyang Lei and Suofei Zhang and Zhou Zhao, “Investigating Capsule Networks with Dynamic Routing for Text Classification”, CoRR, abs/1804.00538, 2018.

[3] Srivastava, Saurabh and Khurana, Prerna and Tewari, Vartika, "Identifying Aggression and Toxicity in Comments using Capsule Network", Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), pp. 98—105, 2018.

[4] Y. Wang, A. Sun, and J. Han, “Sentiment Analysis by Capsules ,” WWW ’18 Proceedings of the 2018 World Wide Web Conference, Pages 1165-1174 vol. 2, 2018.

[5] S. Lai , L. Xu , K. Liu , J. Zhao , Recurrent convolutional neural networks for text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, 333, 2015, pp. 2267–2273.

[6] R. Alan, P. A. Jaques, and J. Francisco, “An analysis of hierarchical text classification using word emb e ddings,” Inf. Sci. (Ny)., vol. 471, pp. 216–232, 2019.

[7] C. Huang , X. Qiu , X. Huang , Text classification with document embeddings, in: Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, Springer, 2014, pp. 131-140 .

[8] B. Wang, W. Liu, Z. Lin, X. Hu, J. Wei, and C. Liu, "Text clustering algorithm based on deep representation learning," vol. 2018, no. Acait, pp. 1407-1414, 2018.

[9] G. Hinton, S. Sabour, and N. Frosst, “MATRIX CAPSULES WITH EM ROUTING,” International Conference on Learning Representations (ICLR), pp. 1–15, 2018.

[10] Jaeyoung Kim, and Sion Jang and Sungchul Choi and Eunjeong Park, “Text Classification using Capsules”, CoRR, abs/1808.03976, 2018.

[11] E. M. Ivan, "Scalable Video Analytics using Capsule Networks for Big Video Data", ScienceDirect, Procedia Comput. Sci., vol. 135, p. 3, 2018.

[12] Duarte, K., Rawat, Y.S., and Shah, M. VideoCapsuleNet: A Simplified Network for Action Detection. CoRR, abs/1805.08162, 2018.