Conditional Neural Networks

Marina Sichlimiri

Diploma Thesis

Supervisor: Konstantinos Blekas

Ioannina, February 2022

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Περίληψη

Η μηχανική μάθηση είναι ένα σημαντικό μέρος του ταχέως αναπτυσσόμενου κλάδου της επιστήμης δεδομένων. Οι αλγόριθμοι εκπαιδεύονται για να δημιουργούν ταξινομήσεις ή προβλέψεις χρησιμοποιώντας στατιστικές προσεγγίσεις, αποκαλύπτοντας κρίσιμες γνώσεις σε πρωτοβουλίες εξόρυξης δεδομένων. Συνελικτικά νευρωνικά δίκτυα, αυτο-εποπτευόμενη μάθηση, προσαρμογή τομέα, ενεργή μάθηση, γραφήματα γνώσης και άλλες εξελιγμένες προσεγγίσεις νευρωνικών δικτύων αναπτύχθηκαν πρόσφατα για να χειριστούν πολυάριθμες πρακτικές δυσκολίες στην επεξεργασία εικόνας. Σε αυτή την εργασία, αναλύουμε ένα νέο είδος νευρωνικών δικτύων, τα υπό όρους νευρωνικά δίκτυα. Το Conditional Neural Network (CLNN) είναι μια προσέγγιση βασισμένη σε συνθήκες για την εκμετάλλευση της διαδοχικής σχέσης του χρονικού σήματος στα πλαίσια. Επιπλέον, το Masked Conditional Neural Network (MCLNN), το οποίο προέρχεται από το CLNN, επιβάλλει μια συστηματική αραιότητα που ακολουθεί ένα μοτίβο που μοιάζει με ζώνη συχνοτήτων για να αξιοποιήσει τη μη συνεχόμενη κατανομή των ενεργειών σε μια αναπαράσταση χρόνου – συχνότητας. Εφαρμόσαμε τόσο το CLNN όσο και το MCLNN στο πρόβλημα της ταξινόμησης εικόνων, χρησιμοποιώντας το σύνολο δεδομένων FASHION MNIST, και έχουμε καταλήξει σε διάφορα συμπεράσματα, συμπεριλαμβανομένης της συνάρτησης ακρίβειας και απώλειας του σετ δοκιμών μας.

Abstract

Machine learning is a crucial part of the rapidly expanding discipline of data science. Algorithms are trained to generate classifications or predictions using statistical approaches, revealing crucial insights in data mining initiatives. Convolutional neural networks, self-supervised learning, domain adaptation, active learning, knowledge graphs, and other sophisticated neural network approaches have recently been developed to handle numerous practical difficulties in image processing. In this paper, we analyze a new kind of Neural Networks, the Conditional Neural Networks. The Conditional Neural Network (CLNN) is a condition-based approach for exploiting the temporal signal's sequential relationship across frames. In addition, the Masked Conditional Neural Network (MCLNN), which is derived from the CLNN, enforces a systematic sparseness that follows a frequency band-like pattern to leverage the non-contiguous distribution of energies in a time–frequency representation. We have applied both CLNN and MCLNN to the problem of image classification, using the Fashion MNIST Dataset, and we have made a variety of conclusions, including the accuracy and the loss function of our testing set.

Keywords: Machine Learning, Neural Networks, Deep Learning, Convolutional Neural Networks, Conditional Neural Networks, Masked Conditional Neural Networks, Fashion MNIST dataset, Image Classification, Restricted Boltzmann Machine (RBM), Deep Belief Net (DBN), metrics (Accuracy, Loss Function, Percentage of Predictions), Confusion matrix

Contents

Chapter 1. Introduction

1.1 Machine Learning

Machine Learning is a branch of artificial intelligence (AI) that allows computers to learn and improve on their own without having to be explicitly programmed. Machine Learning is concerned with the creation of computer programs that can access data and learn on their own.

The learning process starts with observations or data, such as examples, direct experience, or instruction, so that we can seek for patterns in data and make better decisions in the future based on the examples we provide. The fundamental goal is for computers to learn on their own, without the need for human involvement, and to change their behavior accordingly.

Machine Learning algorithms are used to produce predictions or classifications in general. Your algorithm will generate an estimate about a pattern in the data based on some input data, which can be labeled or unlabeled. The model's prediction is evaluated using an error function. If there are known examples, an error function can be used to compare the model's accuracy. Weights are modified to lessen the difference between the known example and the model estimate if the model can fit better to the data points in the training set. This evaluate and optimize procedure will be repeated by the algorithm, which will update weights on its own until a certain level of accuracy is reached.

The way an algorithm learns to become more accurate in its predictions is how traditional Machine Learning is often classified. Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning are the four basic methodologies. The algorithm that data scientists use is determined by the sort of data they wish to predict. So,

- *Supervised learning:* In this case, algorithms are supplied with labeled training data and the variables are defined to be assessed for correlations. The algorithm's input and output are both provided.
- *Unsupervised learning:* Algorithms that train on unlabeled data are used in this sort of machine learning. The algorithm looks for relevant connections between data sets. The data used to train algorithms, as well as the forecasts or suggestions they produce, are all predetermined.
- *Semi-supervised learning:* This method of Machine Learning combines the two previous approaches. Although algorithm may be fed with a largely labeled training data, the model is allowed to explore the data and establish its own understanding of the set.
- *Reinforcement learning:* Reinforcement learning is a technique used to teach a machine to perform a multi-step procedure with well-defined rules. Data scientists design an algorithm to perform a task and provide it with positive or negative feedback as it figures out how to do so. However, the algorithm, for the most part, selects what actions to take along the road on its own.

If not tackled deliberately, the process of selecting the best machine learning model to solve a problem can be time consuming. So, the problem must be aligned with prospective data inputs that should be considered when developing a solution and the process must be continuing until the outputs reach an acceptable level of accuracy.

Machine Learning has progressed significantly over time. It's now being used in a wide range of businesses, and experts in all aspects of Machine Learning are in high demand. From 1943 to today, this infographic below [Fig. 1.] depicts the evolution of Machine Learning throughout history – including significant achievements and milestones:

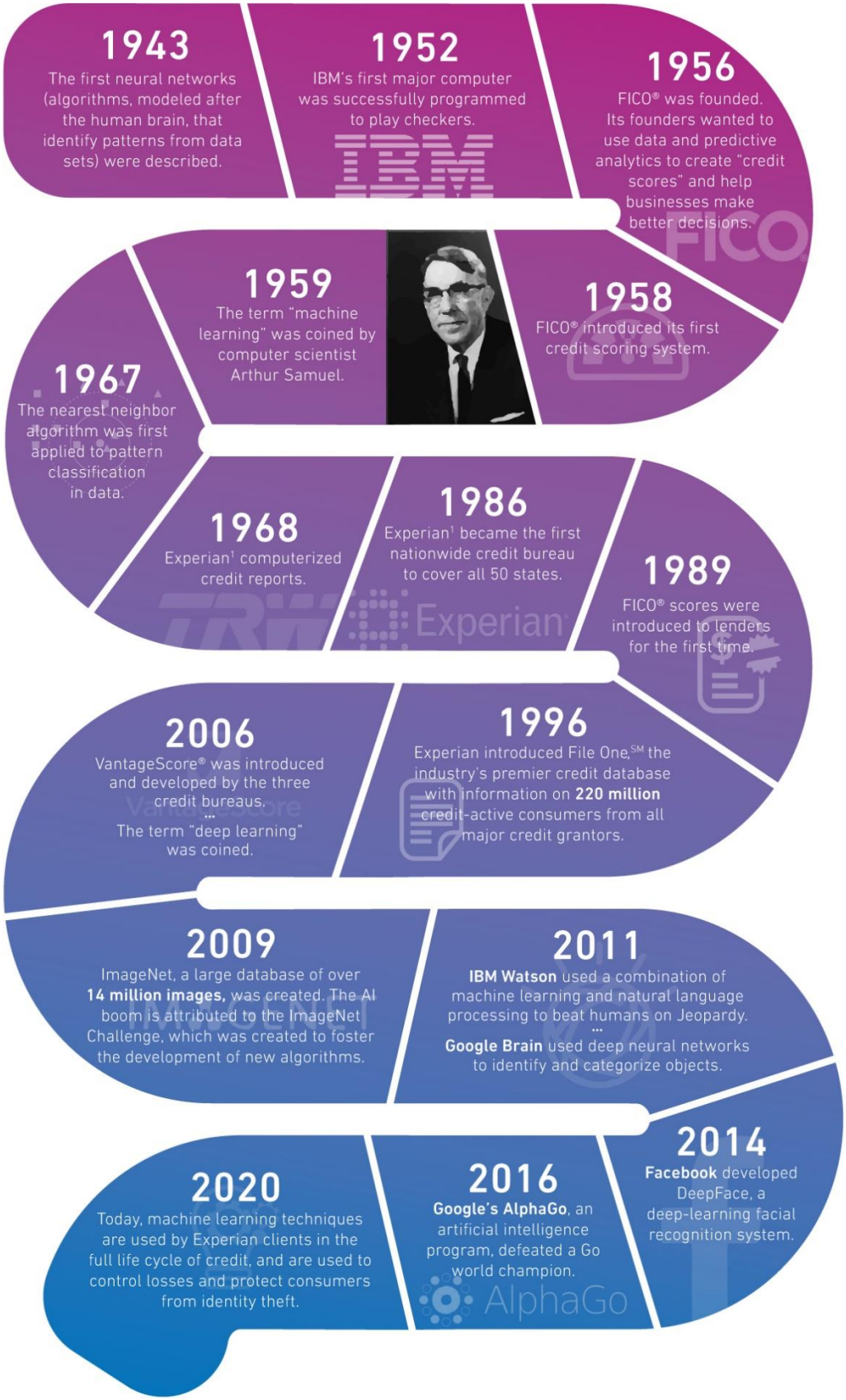


Fig. 1. Machine Learning's History.

While Machine Learning algorithms have been around for decades, their popularity has risen in tandem with the rise of artificial intelligence. Deep Learning models are at the heart of today's most advanced artificial intelligence systems.

The machine learning platform conflicts will only grow as Machine Learning becomes more important to company operations and AI becomes more feasible in enterprise settings. Deep learning and AI research is increasingly focusing on generating more generic applications. To build an algorithm that is highly optimized to accomplish one task, today's AI models require significant training. However, other academics are looking into ways to make models more adaptable, such as techniques that allow a machine to use context acquired from one work to future, unrelated activities.

In Fig. 2., it can be seen clearly that Deep Learning works in very different ways than traditional Machine Learning:

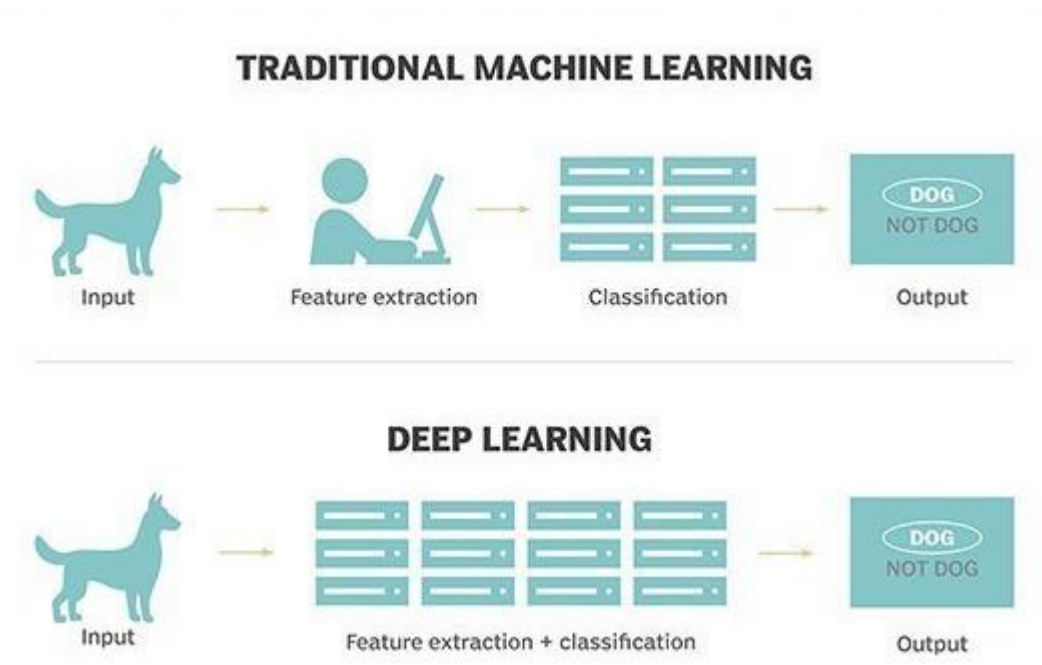


Fig. 2. Comparison between Machine Learning and Deep Learning.

1.2 Deep Learning

Because deep learning and machine learning are often used interchangeably, it's important to understand the differences. Artificial intelligence includes subfields such as Machine Learning, Deep Learning, and Neural Networks. Deep Learning, on the other hand, is a branch of machine learning, and neural networks is a branch of deep learning.

The way each algorithm learns is where deep learning and machine learning differ. Deep Learning automates a substantial portion of the feature extraction process, removing some of the need for manual human involvement and allowing for the usage of bigger data sets. Deep Learning is something like “scalable machine learning” as Lex Fridman [AI researcher] denotes. Classical machine learning, often known as "non-deep" machine learning, is more reliant on human assistance to learn. The set of features used by human experts to interpret the distinctions between data sources is usually determined by more structured data.

Labeled datasets, also known as supervised learning, can be used to inform Deep Machine Learning algorithms, but they aren't always required. It can consume unstructured data in its raw form (e.g., text, images) and determine the set of features that separate different categories of data from one another automatically. It does not require human intervention to interpret data, unlike Machine Learning, allowing us to scale machine learning in more exciting ways. Deep Learning and neural networks are credited for speeding up progress in fields including computer vision, natural language processing, and speech recognition.

Deep Learning achieves higher recognition accuracy than ever before and it has progressed to the point where it now outperforms humans in some tasks, such as categorizing objects in images. Although it was first proposed in the

1980s, it has only lately become relevant due to these two reasons: Deep Learning requires large amounts of labeled data and substantial computing power.

Deep Learning models are sometimes referred to as deep neural networks because most Deep Learning approaches use Neural Network designs. The number of hidden layers in a neural network is commonly referred to as "deep." Deep Neural Networks can have up to 150 hidden layers, whereas traditional neural networks only have 2-3. Large sets of labeled data and Neural Network topologies that learn features directly from the data without the requirement for manual feature extraction are used to train Deep Learning models. For further information, Neural Networks will be examined thoroughly in next section.

Regarding to what can be considered as a difference between Deep Learning and Machine Learning, Deep Learning is a type of machine learning that is very specialized. The first step in a Machine Learning workflow is to manually extract useful characteristics from images. The features are then utilized to build a classification model for the items in the image. Relevant features are automatically retrieved from images using a Deep Learning approach. Furthermore, deep learning accomplishes "end-to-end learning," in which a network is given raw data and a task to complete, such as classification, and it automatically learns how to do so. Another significant distinction is that Deep Learning algorithms scale as data grows, whereas shallow learning techniques converge. Machine Learning algorithms that plateau at a given level of performance as more instances and training data are added to the network are referred to as shallow learning. Furthermore, Deep Learning networks have the advantage of continuously improving as the size of your data grows. To sort images using Machine Learning, you must manually select features and a classifier, whereas the steps of feature extraction and modeling are automated with deep learning.

Fig. 3. denotes the process used by two techniques, Machine Learning and Deep Learning, in a vehicle classification. Here, Deep Learning technique uses a very known and significant kind of Neural Networks, Convolution Neural Network, which will be analyzed further in next section.

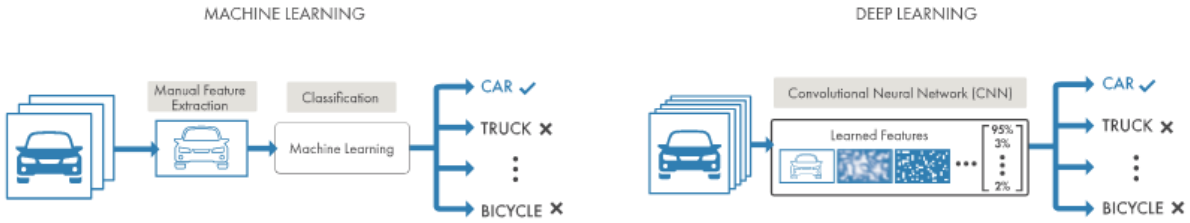


Fig. 3. Comparison between Machine Learning and Deep Learning.

1.3 Neural Networks

Neural Networks, or Artificial neural networks (ANNs) more officially, are made up of node layers that include an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, is connected to the others and has a weight and threshold linked with it. If a node's output exceeds a certain threshold value, the node is activated, and data is sent to the next tier of the network. Otherwise, no data is sent on to the network's next tier. The term "deep learning" simply refers to the number of layers in a neural network. A deep learning algorithm or a deep neural network is a neural network with more than three layers, including the inputs and outputs. A two- or three-layer neural network is referred to as a basic neural network.

In most cases, especially in deep learning, the neurons are grouped into many layers. Neurons in one layer are only connected to neurons in the layers immediately before and following it. The input layer is the one that receives

data from the outside world. The output layer is the one that generates the final product. There are zero or more hidden layers between them. Networks with a single layer and unlayered layers are also employed. Multiple connection patterns are conceivable between two layers. Every neuron in one layer can communicate with every neuron in the next layer, allowing them to be 'completely connected.' Pooling is a technique in which a group of neurons in one layer link to a single neuron in the next layer, lowering the number of neurons in that layer. Feedforward networks are made up of neurons that have only these connections and create a directed acyclic graph. Recurrent networks, on the other hand, are networks that allow connections between neurons in the same or prior levels.

Neural networks learn (or are trained) by analyzing examples with a known "input" and "output," establishing probability-weighted associations between the two that are stored within the net's data structure. The difference between the network's processed output (typically a prediction) and a target output is frequently determined when training a neural network from a given sample. This discrepancy is the flaw. The network then modifies its weighted associations using this error value and a learning strategy. With each change, the neural network will create output that is more and more comparable to the goal output. The training can be ended based on specified criteria after enough of these modifications have been made - this is known as supervised learning which is analyzed in previous section.

These systems "learn" to execute tasks by looking at examples, rather than being coded with task-specific rules. They might learn to identify photographs that contain cats, for example, by studying sample images that have been manually classified as "cat" or "no cat" and then applying the results to detect cats in other images. They achieve this without having any prior knowledge of cats; instead, they use the instances they process to generate identifying traits.

Learning is the process of a network adapting to a task by considering sample observations. To improve the accuracy of the outcome, the weights (and optional thresholds) of the network are adjusted. This is accomplished by reducing the observed mistakes. When looking at more observations does not help reduce the error rate, the learning is complete. Even after learning, the error rate rarely falls below 1%. If the error rate is too high after learning, the network must usually be rebuilt. In practice, this is accomplished by creating a cost function that is reviewed on a regular basis during the learning process. Learning will continue if its output declines. Frequently, the cost is defined as a statistic that can only be approximated. Because the outputs are numbers, the difference between the output (almost likely a cat) and the right answer (cat) is modest when the mistake is low. The goal of learning is to lower the overall number of disparities between observations. Most learning models are simple applications of optimization theory and statistical estimation.

Another crucial parameter is the learning rate, which determines how large the model's corrective steps are in adjusting for faults in each observation. A high learning rate reduces training time but reduces overall accuracy, whereas a low learning rate takes longer but has the potential for greater accuracy. Also, it cannot be omitted the method of backpropagation. Backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input–output example for a single input–output example, and it does it quickly, unlike a naive direct computation of the gradient with respect to each weight individually. Gradient descent, or variants such as stochastic gradient descent, are often used for training multilayer networks, updating weights to minimize loss, because of their efficiency.

In the figures below there are some paradigms of different kinds of Neural Networks analyzed before: an Artificial Neural Network (ANN) – basic Neural Network [Fig. 4.], a Deep Neural Network [Fig. 5.] and a Recurrent Neural Network [Fig. 6.].

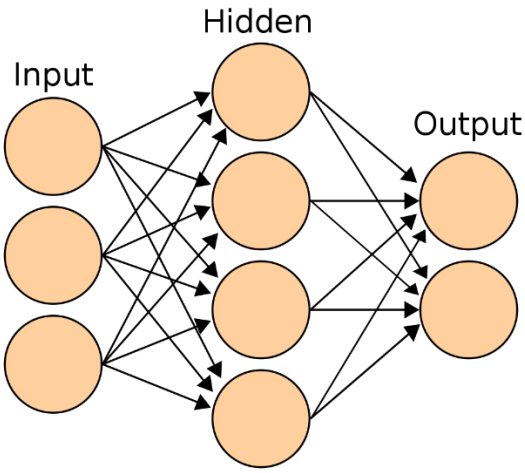


Fig. 4. An Artificial Neural Network (ANN) with one hidden layer.

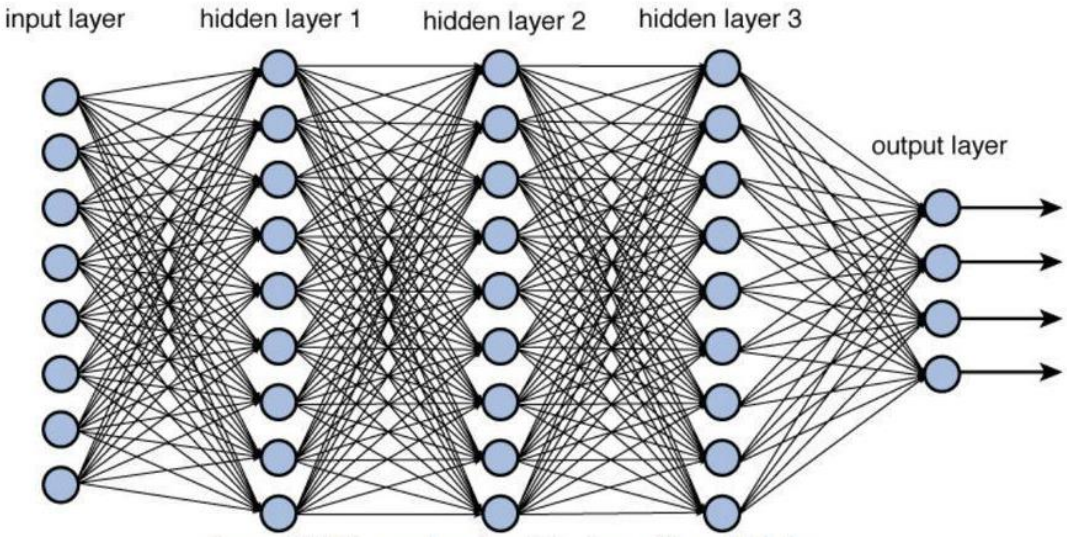


Fig. 5. A Deep Neural Network with three hidden layers.

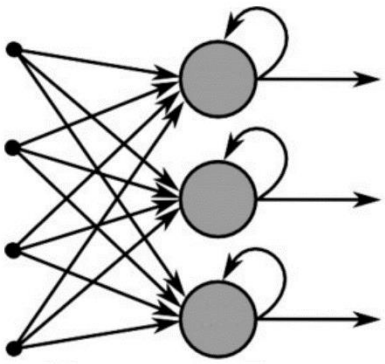


Fig. 6. A Recurrent Neural Network.

1.4 Thesis Overview

Having understood all the above and having learned the current technological developments related to the science of artificial intelligence, in this report we will try to analyze a new type of neural network, which is under study, the Conditional Neural Networks, which are a combination of two types of already known techniques with a wide range of applications, the Convolutional Neural Networks and the Decision Trees. Their expansion, the Masked Conditional Neural Networks, will also be considered, as they operate largely in the same way, but they have added the concept of a mask, utilizing only useful information. In this endeavor, we will have a series of parameters which will play a key role in our experiments and results. Changing their values - and particularly in bandwidth and overlap - we are at the advantage of creating another variation regarding the mask matrix, the Rectangle Mask Matrix. So, we have tried new experiments and thus have come to new conclusions and results. All these are analyzed in detail in the sections following: Section 2 includes the background of Conditional Neural Networks, the ancestors which had been the reason for Conditional Neural Networks to develop. In Section 3 and 4, there is the analysis of Conditional Neural Networks and Masked Conditional Neural Networks respectively. There are shapes, mathematical relations and plain text that explain their structure and their functionality. In the end of each of two Sections, there is our approach for these two models. The results of all our simulations and experiments are in Section 5 and there is a conclusion (and bibliography) in Section 6 which summarizes all the above.

Chapter 2. Background

2.1 A Brief Backward Glance

There have been successful non-speech classification systems since the 1990s, and possibly before. For discriminating between distinct sound categories, the majority relied on hand-crafting the most conspicuous acoustic traits. With the success of deep neural architectures, there has recently been a push to reduce the need for hand-crafting categorization features. Deep architectures are applied directly to the raw signal or an intermediate representation, such as spectrograms, to achieve this. In most cases, hand-designed features still win out, although the margin is closing as deep learning progresses.

A lot of researchers contributed to all this evolution. Soltau's work is notable for being one of the first to apply neural network designs to audio feature extraction rather than direct categorization. Hamel developed a similar advanced strategy to music feature learning by using stacked Restricted Boltzmann Machine (RBM) architecture to construct a Deep Belief Net (DBN) architecture. Cakir recently attempted to categorize different sound sources overlapping a temporal instance in an outdoor sound scene using a deep neural network architecture. Convolutional Neural Networks (CNN) have also been researched for various music tasks in, and due to a lack of large labeled sound datasets for training, novel CNN models were created with the goal of bringing the CNN advances in image processing to sound.

As previously mentioned, various neural network-based sound experiments have been made. Regardless of the genre of sound (music, speech, or general

environmental noises), there is a lot of overlap between the approaches for feature extraction and recognition. In this section, we go over the most important models for this project in further depth.

Taylor's Conditional Restricted Boltzmann Machine (CRBM), a modification of the RBM, is designed to take into account the temporal signal's sequential link between feature vectors. In addition to the undirected links between the visible and hidden nodes depicted in the figure by W, the CRBM architecture includes directed links (forming the conditional relation) from the previous visible vectors v_{-n}, …, v_{-2}, v_{-1} (feature vectors) to both the hidden layer h and the current visible input vector v_0.

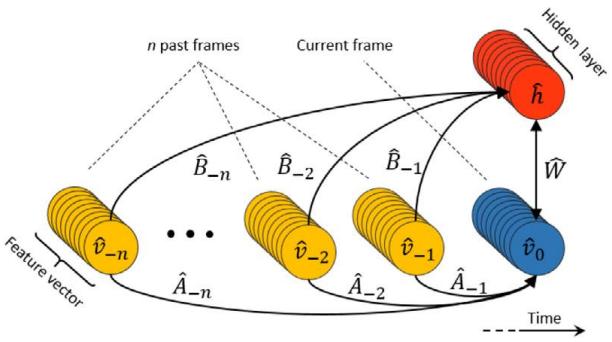


Fig. 7. Conditional Restricted Boltzmann Machine.

The Interpolating CRBM (ICRBM), a CRBM extension, was used for phoneme classification in speech recognition. Because it incorporates the influence of future frames in addition to prior ones, the ICRBM outperforms the CRBM. The models in this paper are based on both the CRBM and the ICRBM. However, they are generative models with several temporal phases that behave like a dense network. When applied to sound, they are given numerous frames of a spectrogram, ignoring the dispersion of energy across the frequency bins of a spectrogram or any two-dimensional representation.

As far as Recurrent Neural Networks' history is concerned, Long Short-Term Memory (LSTM) is a sequence labeling recurrent neural network (RNN) that was developed to address RNN's vanishing and exploding gradient issues. When learning dependencies across a large temporal frame (backpropagation through time), certain issues arise in the RNN. The RNN includes a feedback loop that considers the output state of a sequential signal's prior input with the current input. The LSTM improved this behavior by preserving the state with an internally controlled memory cell. In handwriting recognition, LSTM has reached ground-breaking results. Graves utilized it in his deep LSTM-RNN architecture for speech phoneme recognition. The LSTM's success in text recognition goes much beyond its use to visual or sound signals. This is due to the added complexity and training problems introduced by the memory cell's additional weights, particularly with the higher number of characteristics (frequency bins) in sound compared to text (literal mapped to numerical indices). This prompted efforts like Choi's, who introduced a Convolutional RNN (CRNN) for music classification tagging. A CNN and an LSTM are combined in their CRNN. They employed a CNN to summarize the local properties of a spectrogram, which were then fed into an LSTM to take advantage of the long-term dependencies between frames.

2.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a type of artificial neural network used to interpret visual imagery in deep learning. Based on the shared-weight architecture of the convolution kernels or filters that slide along input features and give translation equivariant responses known as feature maps, they are also known as shift invariant or space invariant artificial neural networks (SIANN). Surprisingly, most convolutional neural networks are only equivariant under translation, rather than invariant. Image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series are just some of the areas where they can be used. Multilayer perceptrons are regularized variants of CNNs. Multilayer perceptrons are typically completely connected networks, meaning that each neuron in one layer is linked to all neurons in the following layer. These networks' "complete connectedness" makes them vulnerable to data overfitting. Regularization, or preventing overfitting, can be accomplished in a variety of methods, including punishing parameters during training (such as weight loss) or reducing connectivity (skipped connections, dropout, etc.) CNNs take a different approach to regularization: they take advantage of the hierarchical pattern in data and use smaller and simpler patterns imprinted in their filters to assemble patterns of increasing complexity.

An input layer, hidden layers, and an output layer make up a convolutional neural network. Any middle layers in a feed-forward neural network are referred to as hidden since the activation function and final convolution mask their inputs and outputs. The hidden layers of a convolutional neural network include convolutional layers. This usually comprises a layer that does a dot product of the convolution kernel with the input matrix of the layer and its activation function is commonly ReLU. The convolution procedure generates a feature map as the convolution kernel slides along the input matrix for the

layer, which then contributes to the input of the following layer. Other layers such as pooling layers, fully connected layers, and normalization layers are added after this.

In CNNs, the input is convolved by convolutional layers, which then pass the output to the next layer. Input's shape is a tensor as following: (number of inputs) x (input height) x (input width) x (input channels). The image is abstracted to a feature map, also known as an activation map, after passing through a convolutional layer, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels). The input is convolved by convolutional layers, which then pass the output to the next layer. Each convolutional neuron only processes data for the receptive field it is assigned to. Although fully linked feedforward neural networks can be used to learn features and classify data, this architecture is unsuitable for bigger inputs like high-resolution photos. Using regularized weights over fewer parameters eliminates the vanishing gradients and inflating gradients issues that plague standard neural networks during backpropagation. In conclusion, because spatial interactions between different features are considered during convolution and/or pooling, convolutional neural networks are appropriate for data with a grid-like architecture (such as photographs).

Regarding pooling layer, by integrating the outputs of neuron clusters at one layer into a single neuron at the next layer, data dimensions are reduced. Small clusters are combined using local pooling, which typically uses tiling sizes of 2 x 2. The feature map's neurons are all affected by global pooling. There are two types of pooling that are commonly used: maximum and average. The maximum value of each local cluster of neurons in the feature map is used in max pooling, while the average value is used in average pooling. Fig. 8., illustrates the CNN process from an input image to output data. This example depicts an image classification where the input image belongs to a vehicle category. The Convolutional and Pooling Layer can be

clearly distinguished and so as the application of ReLU function. Lastly, image turns into a vector where SoftMax function is applicated and the final output-category is predicted.

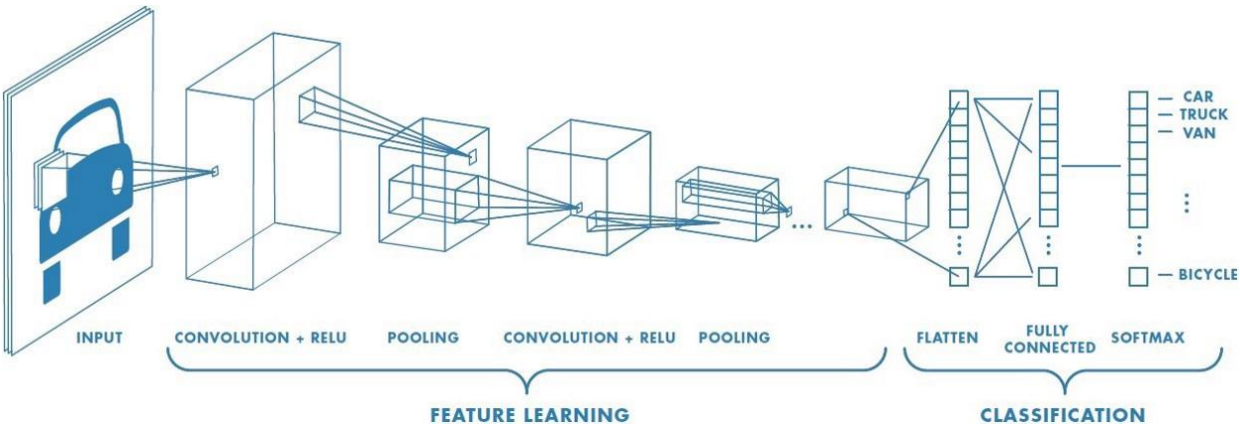


Fig. 8. the CNN's process.

2.3 Decision Trees

The denotative representation of a decision-making process is the definition of a decision tree. In artificial intelligence, decision trees are used to reach conclusions based on the evidence available from previous judgments. Furthermore, these conclusions are given numerical values, which are then used to forecast the course of action that will be done in the future.

Decision trees are statistical, algorithmic machine learning models that interpret and develop responses to a variety of problems and their potential outcomes. As a result, based on the data available, decision trees understand the rules of decision-making in certain settings. The learning process is ongoing, and feedback driven. This enhances the learning outcome over time. Supervised learning is the term for this type of learning. As a result, decision tree models serve as aids to supervised learning.

The decision's tree type is determined by the type of target variable we have. There are two different kinds of it: Categorical Variable Decision Tree (where Decision Tree has a categorical target variable) and Continuous Variable Decision Tree (where Decision Tree has a continuous target variable).

The decision tree algorithm, as the name suggests, has a tree-like structure. It is, however, inverted. A decision tree classifies data sets based on the values of carefully selected qualities, starting at the root or top decision node. The complete dataset is represented by the root node. The best predictor variable is chosen in the first stage of the method. It becomes a decision node because of this. It also divides the entire dataset into several classes or subsets.

A common example of classification is determining if a given image is of a cat or a dog. The presence of claws or paws, the length of the ears, the type of

tongue, and so on are examples of traits or attributes. Based on these input variables, the dataset will be further divided into smaller classes until the desired result is attained.

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example. Each node in the tree represents a test case for some property, with each edge descending from the node corresponding to the test case's possible solutions. This is a cyclical procedure that occurs for each subtree rooted at the new node.

The decision to make strategic splits has a significant impact on a tree's accuracy. The decision criteria for classification and regression trees are different. To decide whether to break a node into two or more sub-nodes, decision trees employ a variety of techniques. The homogeneity of the generated sub-nodes improves with the generation of sub-nodes. To put it another way, the purity of the node improves as the target variable grows. The decision tree divides the nodes into sub-nodes based on all available variables, then chooses the split that produces the most homogeneous sub-nodes.

In conclusion, Decision Trees are classic and natural learning models. They are founded on the basic principle of divide and conquer. Decision trees are used to teach learning computers how to assess success and failure in the field of artificial intelligence. The data is subsequently analyzed and stored by these learning machines. So, they make a slew of decisions depending on their prior knowledge. These judgments serve as the foundation for predictive modeling, which aids in the prediction of problem consequences.

2.4 Ensemble Methods

Ensemble methods are meta-algorithms that combine numerous machine learning techniques into a single predictive model to reduce variance (bagging), bias (boosting), or increase prediction accuracy (stacking).

Ensemble methods are machine learning methods that aggregate the results of multiple predictive models into a single prediction. The goal of merging numerous models is to improve prediction performance, and ensembles have been found to be more accurate than single models in several circumstances. While some work on ensemble methods was done in the 1970s, it wasn't until the 1990s, when approaches like bagging and boosting were introduced, that ensemble methods became more commonly used. They are now regarded as a standard machine learning strategy that must be taken into account whenever high predicted accuracy is required.

When machine learning and advanced learning methods have grown in popularity, ensemble approaches have become increasingly popular. Because ensemble approaches repeat the procedure many times, the model learns the data and generates correct predictions more quickly than when the model uses bagging or boosting, it has a bigger impact on creating correct predictions than when the model does not use it. When Linear Regression, Logistic Regression, or Decision Trees are employed, they only train once on the entire set, but ensemble approaches perform the task many times, similar to how people become experts in any activity after doing it many times.

Random Forest is just the group of Decision Trees combined to give one output or result. Decision Trees, which are analyzed before, are combined to form Random Forest which uses a bagging technique which is Bootstrapping and Aggregating.

Simply said, Bootstrapping is the process of collecting a sample from the data with replacement and then combining all of the samples from the population. Bagging is an ensemble technique in which various samples are collected in order to make a judgment. In Random Forest, we select such samples from the population and create multiple decision trees, not just one. Many trees (bootstrap samples) are joined to make Random Forest. Random Forest's forecast is calculated as the average of all the predictions made by the 'n' number of decision trees. Parallel learning is the name given to this sort of assembly, whereas boosting use sequential learning. As the data is trained on different samples, the independence between the base learners (each decision tree) is established, and all of the scores are calculated to determine the best score. The term "sequential" denotes that there is a relationship between base learners, which is employed by all boosting techniques. Random Forest solely use the bagging technique to reduce the dataset's volatility. Bagging aids in the reduction of variation in data because as the number of decision trees increases, the learning rate increases, resulting in well-trained data and a reduced risk of overfitting. Bagging is a technique for lowering overall variance by combining the outputs of many classifiers trained on distinct samples of the training data.

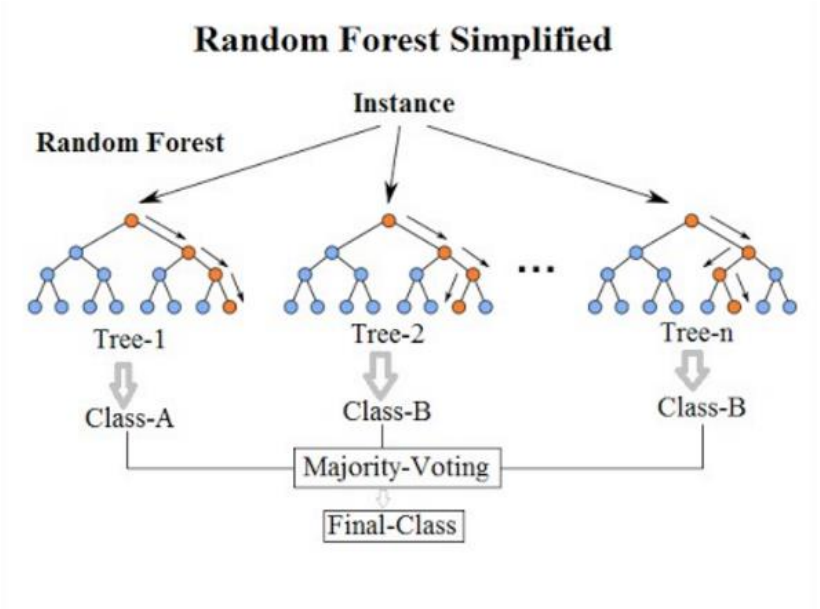


Fig. 9. Random Forest diagram.

Chapter 3. Conditional Neural Networks (CLNNs)

3.1 Introduction

The study provides a series of models called Conditional Neural Networks (CLNNs) that are hybrids of Decision Forests and Neural Networks, as the title suggests. Because of its conditional computation characteristic, Decision trees are computationally efficient (computation is confined to only a small region of the tree, the nodes along a single branch). On the other hand, CNNs, thanks to their representation learning capabilities, attain state-of-the-art accuracy. We can modify the hyper-parameters to generate the model of our choice, whether we want it to be more efficient or more accurate, with the continuous set of models described in the paper.

CLNNs can be viewed in two ways. One is decision trees augmented with data transformation operators and the other one is CNNs, with block-diagonal sparse weight matrices, and explicit data routing functions. Let's start with the first method of visualization. They are Decision Trees with the exception that instead of delivering the input as is, each node applies a non-linear transformation to it. We can also forward the data to one or more of its children using routers. The opposite point of view is that we regard the entire model as a neural network, with block-vise connected layers instead of completely connected layers.

Let's look at a couple of hyper-parameters in more detail. The first is the number of children per node (n), and the second is the number of children to whom the data will be distributed (m). If m = n, the model will act more like a traditional Neural Network, with better accuracy and more training time.

Setting m to a small number, such as 1, will result in a model that is faster but less accurate. So, depending on the application, we can fiddle with these hyper-parameters to get the model that suits us best. Also, as previously said, we can have both stochastic and hard routing depending on the application.

3.2 Analysis of CLNN

The Conditional Neural Network (CLNN), we show in this thesis, is the main structure over which the mask in our Masked Conditional Neural Network (MCLNN), described in the next chapter, is applied.

The CLNN, like other previously proposed temporal models, exploits interframe interactions over a window of frames. Conditional connections, analogous to directed connections from the visible to the hidden layer of the generative CRBM, are used in the CLNN to produce the windowing behavior. It considers future frames as well as prior ones, as the ICRBM has shown.

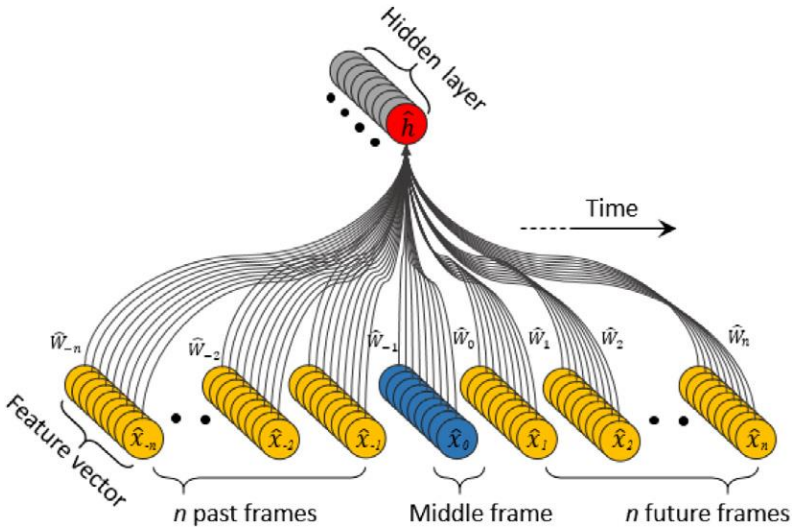


Fig. 10. Conditional Neural Network layer. Connections depicted are for a single hidden node.

A single CLNN layer is shown in Fig. 10. The input consists of d frames (x_{-n}, …, x_{-2}, x_{-1}, x_0, x_1, x_2, …x_n), each with a size of l (i.e. l features). The window's middle frame is x0, which is compared to n (referred to as the order) frames on either temporal axis. Each feature in any frame has a dense connection to each neuron in the hidden layer. For clarity, the picture displays the dense connection for a single neuron, while each neuron in the hidden layer and each of the feature vectors in the window have similar connections. The window's d frame count is as follows:

$$d = 2n + 1, n \geq 1 \quad (1)$$

where n is the number of items in the order. As a result, for the middle frame of a window, the observed pattern of the activation vector at the hidden layer is conditioned on n previous and n subsequent frames.

A single activation vector is generated by a single temporal step across d frames using a CLNN. As a result, a deep CLNN architecture's input frames must account for the reduction in the number of frames at each layer. There must be q frames following the input segment:

$$q = (2n)\, m + k, \qquad n, m, \text{and } k \geq 1 \quad (2)$$

where n is the order, m denotes the number of layers, and k denotes the number of extra frames These extra frames are not included in the CLNN layers, which can be flattened or globally pooled to form a single vector, as mentioned for images. Fig. 11. shows the relative sizes of the spectrogram, the segment, and the window.

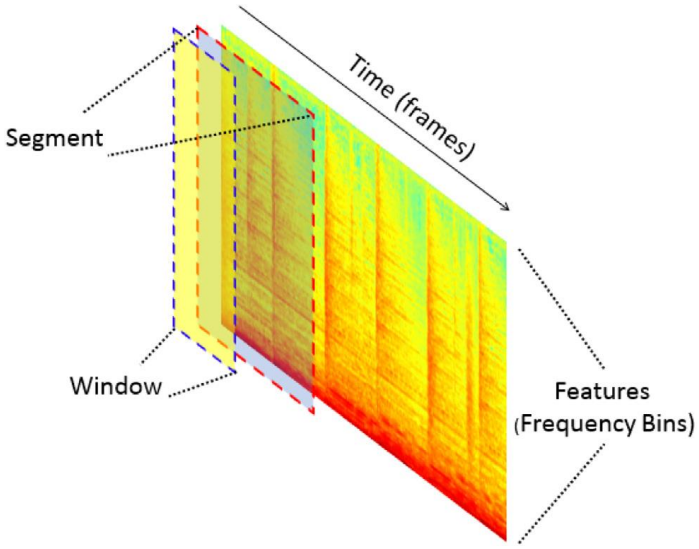


Fig. 11. A CLNN Segment of frames relative size to a Window.

The architecture of a two-layered CLNN (m = 2) with order n = 1 is shown in Fig. 12., where each middle frame in a window is analyzed with one previous frame and one subsequent frame.

As a result, each layer has a three-dimensional weight tensor W_b, with b = 1, 2, ..., m. The weight tensor has a depth of 3 for an order of n = 1. As a result, there is a specialized weight matrix with the same index u for each of the frames within a window (3 frames at n = 1) to process the frame using vector–matrix multiplication. As a result, W_0^b stands for the window's center frame at u = 0, and W_{-1}^b and W_1^b stand for the window's off-center frames at u = -1 and u = 1, respectively. The first CLNN layer transmits q 2n frames to the second CLNN layer, which generates another representation for subsequent layers in the same way. The final output of these two layers is one or more representative frames at the output of the second CLNN (depending on the k additional frames), which can be flattened or pooled across before being fed to a densely connected network before being sent to the final SoftMax layer for classification.

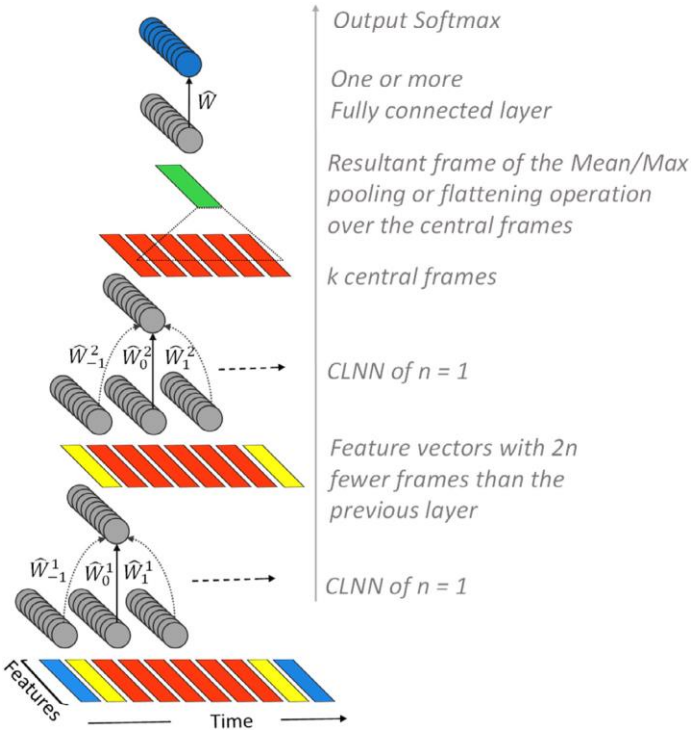


Fig. 12. A two-layer CLNN model with n = 1.

When the CLNN examines a frame at index t within a segment, it considers the middle frame of the window as well as its nearby 2n frames (the middle frame of a window, at u = 0, is also the frame at index t of the segment). Depending on the stride size, successive windows may overlap. The activity of a single buried layer neuron is expressed in (3):

$$y_{j,t} = f\left(b_j + \sum_{u=-n}^{n}\sum_{i=1}^{l} x_{i,u+t} W_{i,j,u}\right) \tag{3}$$

where $y_{j,t}$ represents the hidden layer's output from neuron j, and t is the frame index within segment q. The bias at the hidden neuron is b_j and the activation function is f. Feature I of the vector x_{u+t} is denoted by the phrase $x_{i,u+t}$. Each

feature in a vector of length l is multiplied by the $W_{i,j,u}$ weight element of the matrix Wu. The relationship between the i_{th} feature in the feature vector and the j_{th} hidden node is represented by the indices I j]. [n, n] is the range of the index u. As a result, the number of weight matrices in a window is equal to the number of frames. The hidden layer's output, expressed as a vector, is given in (4):

$$\hat{y}_t = f\left(\hat{b} + \sum_{u=-n}^{n} \hat{x}_{u+t} \cdot \hat{W}_u\right) \tag{4}$$

where y_t is the activation of the segment q regarding the frame at index t. The bias vector b and the transfer function f. The vector at index u in a window is referred to as x_{u+t}. Between the frame at index u within the window d and its corresponding weight matrix at the same index, a vector–matrix multiplication is performed. W_u has the dimensions [l, e], with e being the hidden layer width. A logistic transfer function can be used to capture the conditional distribution, where the hidden layer sigmoid or the final layer SoftMax can be used.

3.3 The CLNN's Proposed Approach

In this subsection, we propose a new Conditional Neural Network architecture being capable of applied to the task of image classification, and more specifically to the known and interesting Fashion MNIST Dataset. In section 3.2, all useful information about CNNs is analyzed and being stuck to this, we give value to some crucial parameters related to this technique, built the layers, run simulations and finally we come up to some results concerning metrics, such as accuracy and loss function.

Firstly, it is important to state that we use Keras Functional API. The Keras functional API allows to design more flexible models than the *tf.keras.Sequential API* allows. Models with non-linear topologies, shared layers, and even multiple inputs and outputs are all supported by the Functional API. A deep learning model is typically a directed acyclic graph (DAG) of layers, according to the fundamental notion. As a result, the Functional API allows to create layer graphs. Keras is a high-level neural networks library that is running on the top of TensorFlow, an end-to-end open-source platform for machine learning – also a significant aid to all this analysis. From TensorFlow library, we import *Input()*, a Keras tensor, which is used to instantiate. A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model. As we have imported the input, as a tensor of 784 neurons (input image 28x28) in the input layer, we utilized *Lambda()* Function in order to define the number of neurons we want to use to make dense connections between layers, as Conditional Neural Networks designate. Also, we used *Dense ()* to perform these dense connections. Last but not least, *concatenate()* assisted to make connections between layers – input, hidden and output.

Having loaded the data set, we normalize the data by dividing by 255, so each pixel has a value in the range 0 to 1. After a lot of simulations about the number of hidden layers that have to be built [variable m in relation (2)] we end up to m=3, as the best option. About variable n [in relation (2)], we have considered for each frame (line of image pixels) the previous three and the next three, including the one we refer to. So, n is also 3 after trying a lot of other values and d=7, according to relation (1). Finally, we have considered k=10 for extra frames, which means that in output level we will have an 10x10 image. More thoroughly, we start from the Input Layer with 784 neurons (input image 28x28), and we continue to the First Hidden layer with 484 neurons (an 22x22 image), to the Second Hidden Layer with 256 neurons (an 16x16 image) and to the Output Layer with 100 neurons (an 10x10 image - k=10 as declared before). The Output Layer succeeds the Final Output Layer, where *flatten ()* is used in order to occur the final classification. Between levels, the dimensions are reduced since the most important features of each image are kept through the training process. These features continue to the next level and determine the final classification in one of the certain classes.

The activation function in a neural network is responsible for converting the node's summed weighted input into the node's activation or output for that input. The rectified linear activation function, or ReLU for short, is a piecewise linear function that, if the input is positive, outputs the input directly; else, it outputs zero. Because a model that utilizes it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks. Although several papers suggest using sigmoid function, as activation function, we end up to better results by using ReLU function [Fig. 13.]:

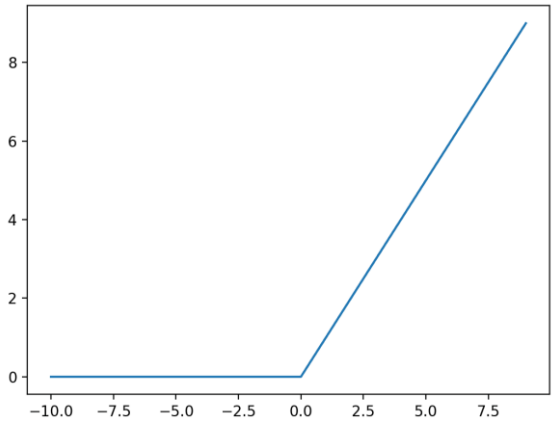


Fig. 13. The Rectified linear activation function – ReLU.

For the final classification SoftMax function [Fig. 14.] is used - a mathematical function that transforms a vector of integers into a vector of probabilities, with the probability of each value proportional to the vector's relative scale. The SoftMax function is most used as an activation function in a neural network model in applied machine learning. The network is set up to produce N values, one for each classification task class, and the SoftMax function is used to normalize the outputs, turning them from weighted sum values to probabilities that total to one. Each number in the SoftMax function's output is interpreted as the likelihood of belonging to each class.

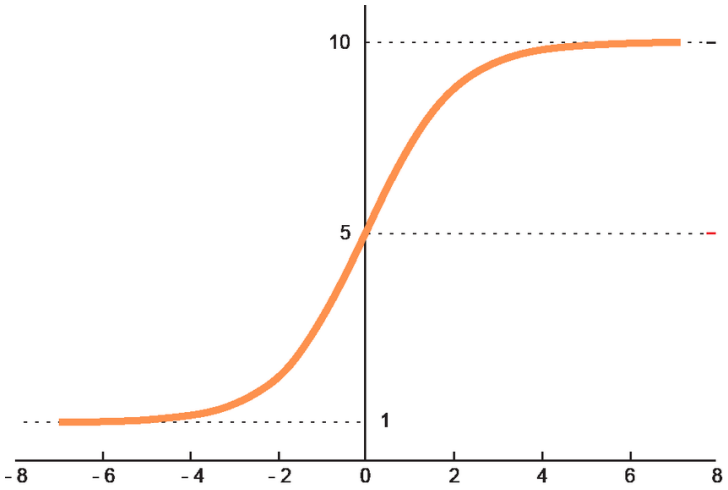


Fig. 14. The SoftMax function.

As far as the Optimizer is concerned, we have selected an Optimizer that implements the Adam algorithm. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. The learning rate is by default 0.001, but we have decreased it to 0.0001. Furthermore, Loss functions are used to determine the quantity that a model should try to minimize during training. This analysis has used *SparseCategoricalCrossentropy* Function as Loss Function as it computes the cross-entropy loss between the labels and predictions. It is commonly used when there are two or more label classes, and we expect labels to be provided as integers.

After all these, we compute some significant metrics so as to evaluate our approach to a Conditional Neural Network – the accuracy, the loss, the percentage of correct predictions of the whole test set and of course the Confusion matrix. Before the representation of the results, the definition of the accuracy and the Confusion matrix must be displayed as they constitute remarkable factors to all this effort:

- Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

- A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes - here is 10. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

The number of epochs is a hyperparameter that controls how many times the learning algorithm runs over the whole training dataset. Each sample in the training dataset has had the opportunity to update the internal model parameters once each epoch. There are one or more batches in an epoch. The batch size is a hyperparameter that specifies how many samples must be processed before the internal model parameters are updated. A batch can be thought of as a for-loop that iterates across one or more samples and makes predictions. The predictions are compared to the expected output variables at the conclusion of the batch, and an error is calculated. The update procedure is employed to improve the model because of this inaccuracy. In this analysis, the batch size is 32, as a result we have 1875 x 32 =60.000 images – the whole training set in each epoch.

Chapter 4. Masked Conditional Neural Networks (MCLNNs)

4.1 Introduction

The MCLNN replicates filterbank-like behavior by enforcing systematic sparseness over the connections between the input and the hidden layer within the network using a binary mask, which extends the structure of the CLNN. As seen in Fig. 15., the mask follows the frequency bands' structural pattern. Two configurable hyperparameters, the Bandwidth (symbolized as bw) and the Overlap (symbolized as ov), regulate the mask construction. The Bandwidth determines the number of features to be examined in a single band (equivalent to a bandpass filter in a filterbank), while the Overlap determines the distance between subsequent bands (mimicking the overlap between filters).

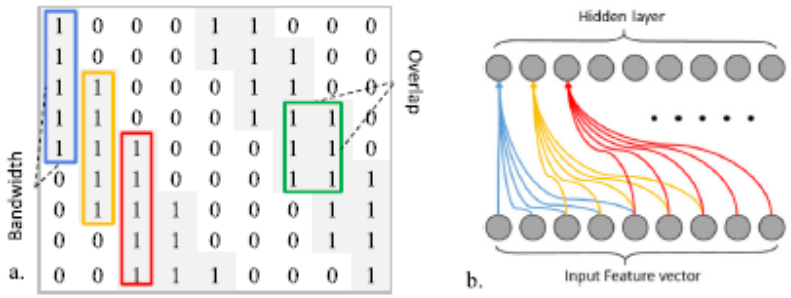


Fig. 15. Examples of the Mask patterns.

In Fig. 15.a., for example, a binary masking pattern with a bandwidth = 5 is represented by the successive ones arranged in a single column. The same masking pattern has an overlap = 3 value, which is represented by the binary

patterns superimposed over successive columns. The enabled links between the input and the hidden layer matching the masking pattern in Fig. 15.a. are shown in Fig. 15.b. Also, negative values for the overlap refer to the non-overlapping distance between the columns.

4.2 Analysis of MCLNN

When a result, as bandwidth grows, the scope of observation of a single hidden node over a feature vector split expands. The linear spacing of the 1's positions within a mask is controlled by the overlap and the bandwidth, as following in (5):

$$lx = a + (g - 1)(l + (bw - ov)) \tag{5}$$

the linear index is lx, the bandwidth is bw, the overlap is ov, and the length of the feature vector is l (number of frequency bins). The values of a and g are within the intervals [1, bw] and [1, (l e)/(l + (bw ov)] respectively.

Through the influence of distinct active regions in the feature vector following the mask design, the filterbank-like pattern causes each neuron in the hidden layer to be activated. Meanwhile, because the active weights' positions are fixed, the spatial localization of the learned characteristics is retained. The systematic sparseness of the input permits connections within a certain band of the input (as if they were frequency bins) to contribute to the hidden node's activation.

Finding the best individual features, as well as the optimal mix of features, is part of hand-crafted features. By embedding altered versions of the filterbank-like pattern in the mask, this procedure is automated. This allows each neuron

to learn about different parts of the feature vector in a unique way. The first neuron in a hidden layer (i.e., the first column in the mask) in Fig. 7.a (with the columns mapping to neurons) will learn about the first five features. Meanwhile, the fifth neuron will learn about the feature vector's first and last two characteristics.

This spatial constraint imposed on the weights, together with shifted versions of the masking pattern, enables selective fusing of the localized distributions of features that contribute to a specific neuron following the shifted binary pattern. This addresses the non-contagious distribution of energy in a time–frequency representation, which the CNN filters are unable to manage. Additionally, the CLNN considers the time component.

A binary mask with the same dimensions as the weight matrix is elementwise multiplied to do the masking operation. This is written as follows in (6):

$$\hat{Z}_u = \hat{W}_u \circ \hat{M} \qquad (6)$$

where Wu is the original weight matrix, M is the masking pattern, and Zu is the new weight matrix after the mask has substituted Wu in the element-wise multiplication (4).

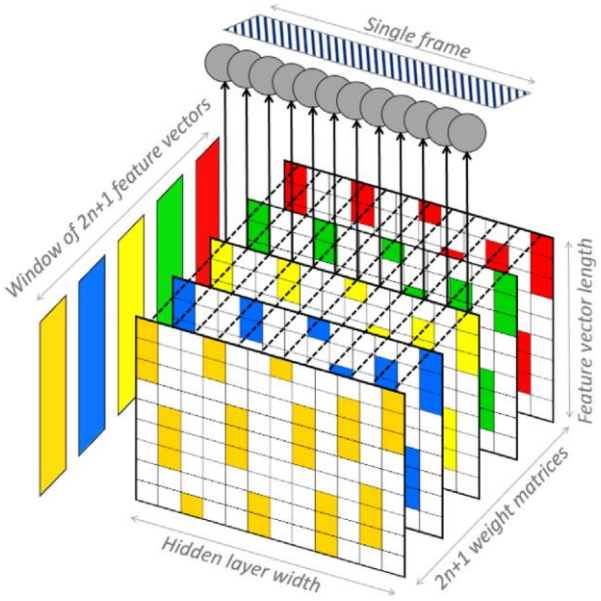


Fig. 16. a single MCLNN step.

As a result, a weight tensor with matrices of count (depth) = 2n + 1 is used to process a window of frames of size 2n + 1. Each frame in the window with the index u is processed with the same index matching matrix. The active connections are depicted in the figure by the highlighted cells. A single resultant frame is the output of an MCLNN step over a window of frames. All these are showed in Fig. 16., where a single step of MCLNN is depicted.

4.3 The MCLNN's Proposed Approach

In this section, we built a Masked Conditional Neural Network using Fashion MNIST Dataset in order to evaluate it. As mentioned before, Masked Conditional Neural Networks make for an extension of Conditional Neural Networks. In section 4.2, it is thoroughly analyzed the role of the mask, which is the main reason why they differ from Conditional Neural Networks.

To begin with, we must specify that we have used Keras Functional API, similarly as CLNN, so as to build our model. From TensorFlow library, we import *Input ()*, *Lambda (), Dense ()* and *concatenate ()* functions to achieve the same structure and functionality of our model. The number of hidden layers is also here three [m=3] and each image of our training set is considered as a vector with 784 neuros again in our Input Layer. After loading the data, we divide it by 255 to make each pixel have a value in the range of 0 to 1 (normalization). As we are proceeding to the next levels, only the useful features of the images are kept, leading to smaller (concerning dimensions' size) but more substantial images. The Final Output Layer uses *flatten ()* function to perform the final categorization among defined classes.

As activation function, we find that utilizing the ReLU function between layers produces better results, but for the final classification among classes, SoftMax function is used again. An Optimizer, which implements the Adam algorithm is presence here, with learning rate 0.0001. The reason why we chose this Optimizer is explained in a previous section (section 5.2).

Following that, we compute the same key metrics to evaluate our Masked Conditional Neural Network's approach: accuracy, loss, percentage of right predictions throughout the entire test set, and, of course, the Confusion matrix. The significance and the functionality of each metric have already been clarified.

To make it clear, Mask matrix is a binary matrix, where the position of ones witness which neuros have to be connected densely (the i_{th} and the j_{th} correspondingly) and, on the other hand, the position of zeros articulates that there will be no connection between neuros. The conformation of the ones and zeros in the matrix is owing to the value of two crucial parameters – bandwidth and overlap. Their value affects greatly the accuracy and the other metrics calculated. This influence is depicted in these plots below:

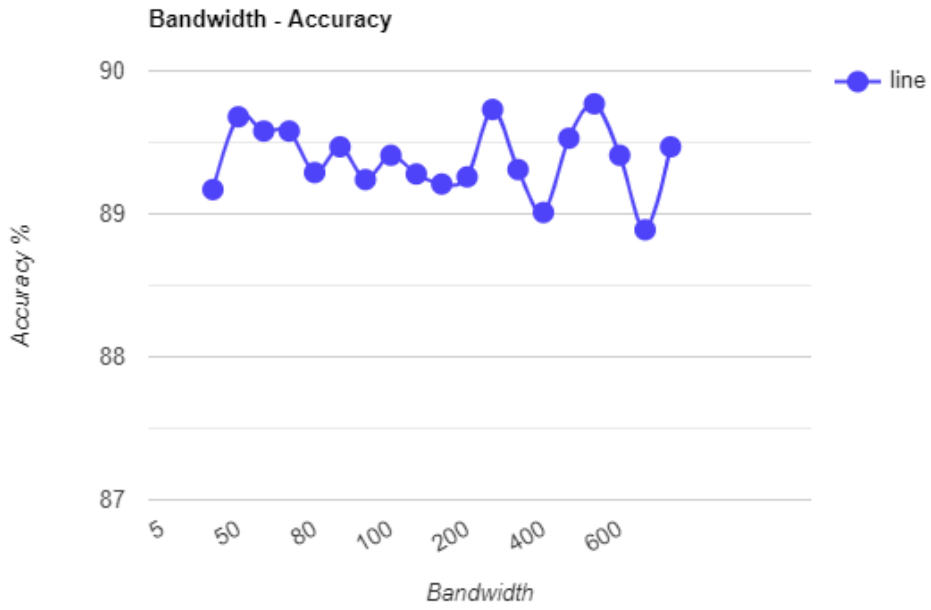


Fig. 17. Bandwidth – Accuracy Plot.

As the bandwidth increases, the value of accuracy has some fluctuations, but all between the [89, 90] interval. If bandwidth is in the middle of [400,600], we can achieve the better (highest) value of accuracy. Keeping bandwidth in this interval, we run simulations about overlap in order to accomplish the best bandwidth - overlap combination which gives us the highest accuracy. So:

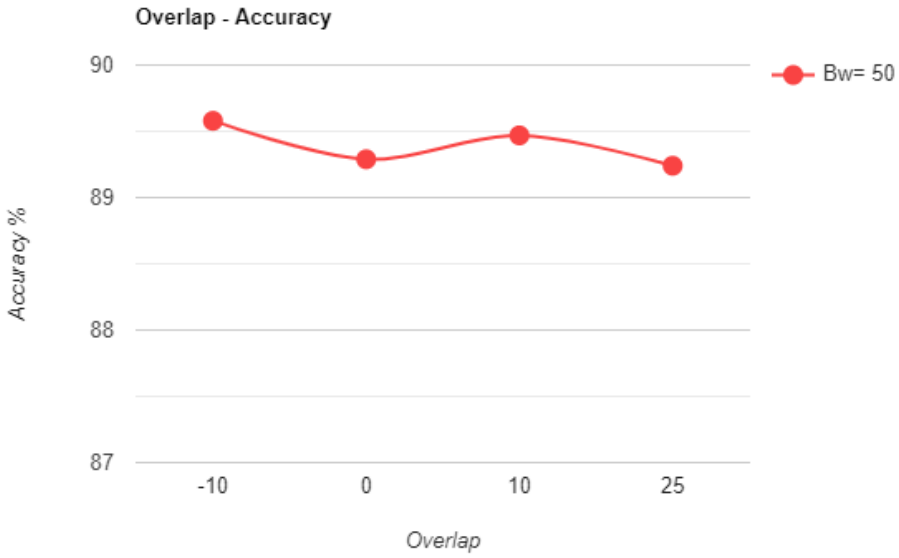


Fig. 18. Overlap – Accuracy Plot with Bandwidth=50.

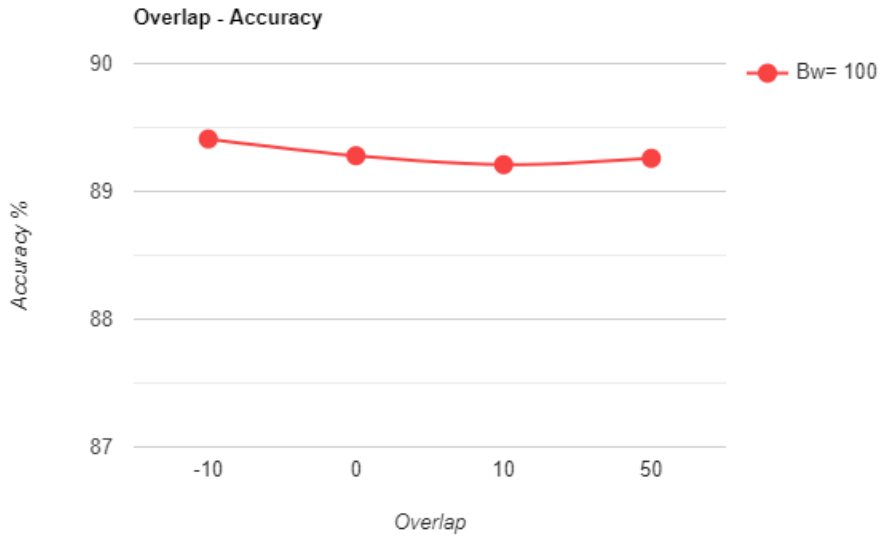


Fig. 19. Overlap – Accuracy Plot with Bandwidth=100.

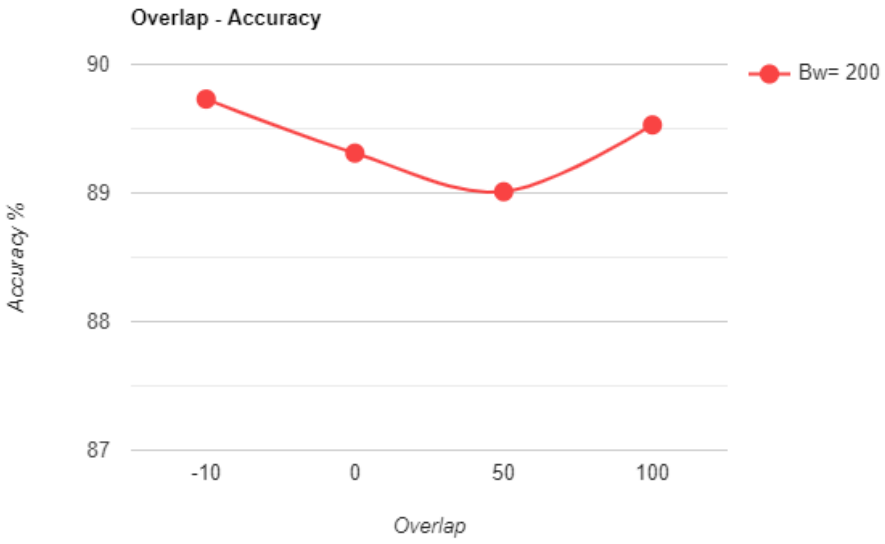


Fig .20. Overlap – Accuracy Plot with Bandwidth=200.

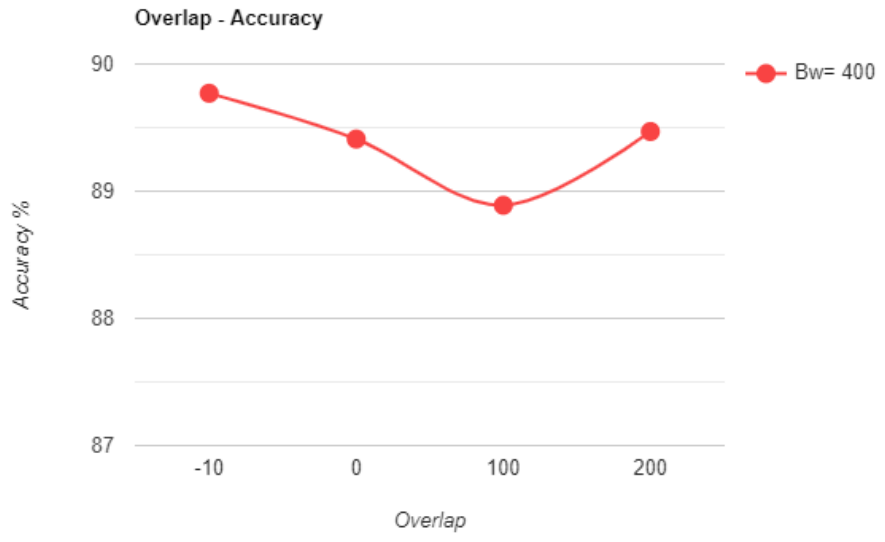


Fig. 21. Overlap – Accuracy Plot with Bandwidth=400.

Having run various simulations and having deliberated all these plots, we ended up to [bandwidth, overlap] = [400, -10] as the best option. Keeping these values stable, we modify the epochs. The number of epochs is a hyperparameter that determines how often the learning algorithm runs across the entire training dataset. The bigger it is, the more times the algorithm learns from the dataset. The batch size is also 32 here and, as a result we have 1875 x 32 =60.000 images – the whole training set in each epoch.

4.3.1. The Proposed Rectangle Mask Matrix schema

By definition, the variables bandwidth and overlap determine how the ones and zeros will be placed at the Mask matrix. In this thesis, we have created a variation concerning the position of ones and zeros in the mask, which has been named Rectangle Mask Matrix. Rectangle Mask Matrix is describing a variation where the ones have placed only at the center of the matrix, covering the 80% of the matrix and the zeros have formed an outline around the ones. As a result, the ones have formed a rectangle in the center of the matrix. This happens because we have paid attention to the form of the Dataset used. Images in FASHION MNIST Dataset gather information in the center, and as the ones of mask matrix witness the nodes which will be taken, we have led to this new different approach. An example is shown next in [Fig. 22.]:

```
[[0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 1 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
```

Fig. 22. A variation of Mask Matrix.

Chapter 5. Experimental Results

5.1 The Fashion MNIST Dataset

The area of image classification has witnessed a revival in the recent decade, thanks to the discovery of deep learning. Newer and more sophisticated deep learning algorithms, such as the convolutional neural network, have supplanted traditional machine learning approaches. To properly comprehend and appreciate deep learning, however, we must first grasp why it succeeds where other methods fail. We use various classification algorithms on the Fashion MNIST dataset to try to answer some of those problems.

Information about the dataset Zalando Fashion's research lab introduced Fashion MNIST in August 2017. Its purpose is to replace MNIST as a new standard for evaluating machine learning algorithms, as MNIST has become too simple and overused. Fashion MNIST is made up of photos of 10 distinct clothing objects, whereas MNIST is made up of handwritten digits. The attributes of each image are as follows:

Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 pixels. Each pixel has a single pixel-value that indicates its lightness or darkness, with larger numbers representing darker pixels. This pixel value is an integer ranging from 0 to 255. There are 785 columns in both the training and test data sets. The first column indicates the article of clothing and contains the class designations. The pixel values of the linked image are contained in the remaining columns. Each training and test example is given one of the labels listed below:

- 0 : T-shirt/top

- 1 : Trouser
- 2 : Pullover
- 3 : Dress
- 4 : Coat
- 5 : Sandal
- 6 : Shirt
- 7 : Sneaker
- 8 : Bag
- 9 : Ankle boot

In Fig. 23., it is a plot of the images in the dataset which are grayscale photographs of items of clothing. Each image belongs to a one and only class declared above:

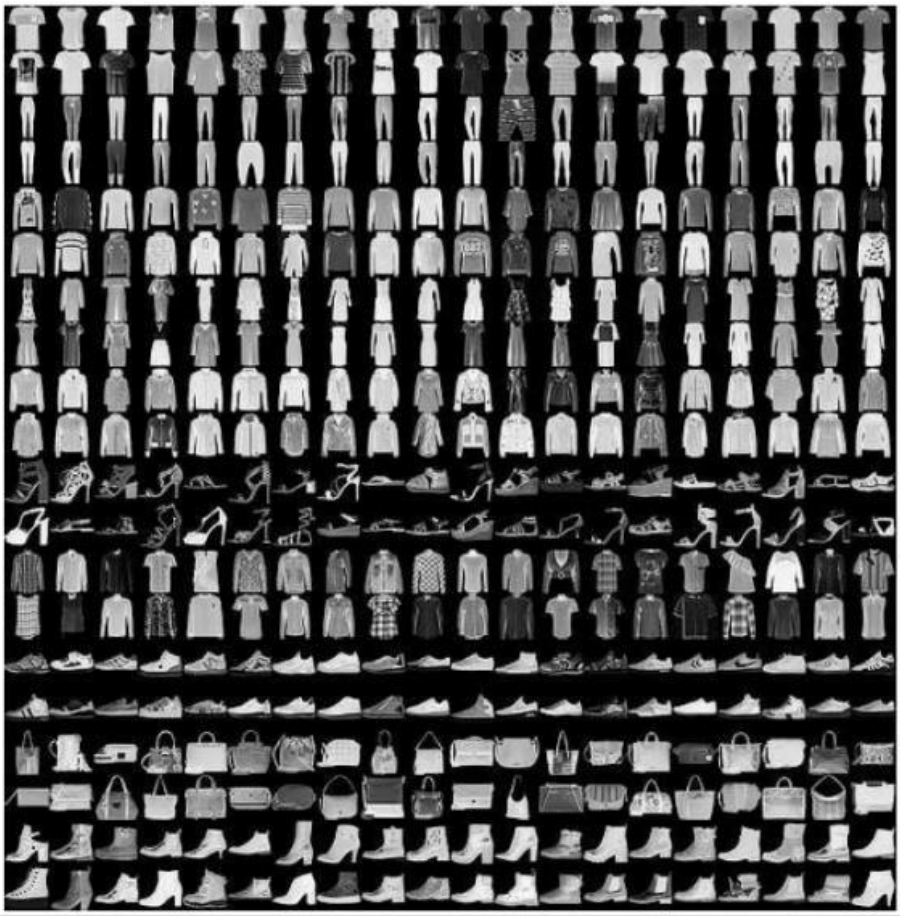
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig. 23. Plot of images from the Fashion-MNIST Dataset.

Convolutional Neural Networks (analyzed in previous section) was the first method used. In fact, it is one of the most basic CNN architectures available. This demonstrates the full potential of this family of methods: achieving excellent results using a benchmark framework. As the images are grayscale, one channel is used.

The algorithm was tested after it converged after 15 epochs, indicating that it is not overtrained. The resulting testing accuracy was 89 percent, which is the greatest result out of all the approaches! CNNs are the most practical and, in most cases, the most accurate method since they can transfer learning across layers, save inferences, and make new ones on following layers. Furthermore,

there is no need to extract features before employing the algorithm because this is done during training. Last but not least, CNNs recognize important features.

They do, however, come with certain drawbacks. They've been known to fail on images that have been rotated and scaled differently, but this isn't the case here because the data has already been processed. And, even if the other approaches don't do as well on this dataset, they're nevertheless employed for other image processing jobs (sharpening, smoothing etc.).

5.2 Results of the CLNN's technique

Having run a lot of simulations, giving the variable epochs the values [25, 50, 100], the results are:

- Epochs = 25

Test accuracy	88.41 %
Test loss	0.324569
Percentage of correct predictions	88.42 %

Table 1. Metrics for CLNN with epochs=25.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	81.3 %	0.1 %	1.5 %	2.7 %	0.4%	0.2%	13.1 %	0	0.7%	0
Class 1	0.5 %	96.9 %	0	2%	0.2%	0	0.3%	0	0.1%	0
Class 2	1.1 %	0.1 %	79.0 %	1.2 %	10.6 %	0.1 %	7.9 %	0	0	0
Class 3	1.5 %	0.6 %	0.9 %	90.1 %	3.0 %	0.2 %	3.4 %	0	0.3 %	0
Class 4	0	0.1 %	7.1 %	2.7 %	84.3 %	0	5.6 %	0	0.2 %	0
Class 5	0	0	0	0.1 %	0	97.2 %	0	1.9 %	0.1 %	0.7 %
Class 6	12.2 %	0.3 %	7.1 %	2.6 %	8.4 %	0	68.5 %	0	0.9 %	0
Class 7	0	0	0	0	0	1.6 %	0	96.1 %	0	2.3 %
Class 8	0.3 %	0.1 %	0.5 %	0.5 %	0.7 %	0.4 %	0.8 %	0.6 %	96.1 %	0
Class 9	0	0	0	0	0	1.2 %	0.1 %	4.0 %	0	94.7 %

Table 2. Confusion Matrix for CLNN with epochs=25.

- Epochs = 50

Test accuracy	88.59 %
Test loss	0.348330
Percentage of correct predictions	88.6 %

Table 3. Metrics for CLNN with epochs=50.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	80.0 %	0.4 %	1.5 %	1.7 %	0.2 %	0.1 %	15.2 %	0	0.9 %	0
Class 1	0.4 %	97.0 %	0	1.6 %	0.2 %	0	0.6 %	0	0.2 %	0
Class 2	1.4 %	0.2 %	81.2 %	0.9 %	7.4 %	0.1 %	8.4 %	0	0.4 %	0
Class 3	1.8 %	0.4 %	1.0 %	87.6 %	2.9 %	0	5.9 %	0	0.4 %	0
Class 4	0.1 %	0.1 %	7.7 %	3.1 %	78.1 %	0	10.4 %	0	0.5 %	0
Class 5	0	0	0	0.1 %	0	95.6 %	0.2 %	2.5 %	0.1 %	1.5 %
Class 6	8.2 %	0.2 %	5.9 %	2.6 %	4.8 %	0	77.4 %	0	0.9 %	0
Class 7	0	0	0	0	0	1.1 %	0	96.1 %	0.1 %	2.7 %
Class 8	0.3 %	0.2 %	0	0.5 %	0.3 %	0.2 %	1.2 %	0.4 %	96.9 %	0
Class 9	0	0	0	0	0	0.6 %	0.1 %	3.2 %	0	96.1 %

Table 4. Confusion Matrix for CLNN with epochs=50.

- Epochs = 100

Test accuracy	88.75 %
Test loss	0.490789
Percentage of correct predictions	88.75 %

Table 5. Metrics for CLNN with epochs=100.

and **Confusion Matrix** is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	83.9 %	0.6 %	1.7 %	2.1 %	0.5 %	0.2 %	10.5 %	0	0.5 %	0
Class 1	0.3 %	97.3 %	0.1 %	1.6 %	0.4 %	0	0.2 %	0	0.1 %	0
Class 2	2.0 %	0.2 %	76.4 %	0.7 %	12.8 %	0.1 %	7.5 %	0	0.3 %	0
Class 3	2.8 %	0.5 %	1.0 %	88.2 %	3.9 %	0	2.9 %	0.1 %	0.6 %	0
Class 4	0.3 %	0	5.3 %	2.2 %	86.9 %	0	4.9 %	0	0.4 %	0
Class 5	0	0	0	0	0	95.8 %	0.1 %	2.4 %	0.3 %	1.4 %
Class 6	12.1 %	0.1 %	5.0 %	1.5 %	9.5 %	0.1 %	70.4 %	0	1.3 %	0
Class 7	0	0	0	0	0	1.4 %	0	95.8 %	0.1 %	2.7 %
Class 8	0.6 %	0	0	0.1 %	0.6 %	0.3 %	0.8 %	0.5 %	97.1 %	0
Class 9	0.1 %	0	0	0	0	1.1 %	0	3.1 %	0	95.7 %

Table 6. Confusion Matrix for CLNN with epochs=100.

As the number of epochs are increasing, we can notice that the accuracy increases as well. This means that our Conditional Neural Network runs over the whole training dataset more and more times, producing better and more accurate predictions.

Also, it is important to clarify that as the variable of epochs increases, the loss follows the same case –we wish it didn't happen. It's possible that this is due to overfitting. This can happen as the model continues to make incorrect predictions (on the test set) with increasing confidence. This means it doesn't make more incorrect predictions with time (thus the stable/increasing accuracy), but it becomes increasingly confident in its existing incorrect forecasts (explaining the increasing cost).This could be due to the model overfitting on training data characteristics that do not generalize to test data. This is especially true for MNIST datasets, where "spurious" features (such as single pixels) are frequently overfitted.

Confusion Matrices witness that in each class we have a very satisfactory number of correct predictions, however there are little wrong predictions in each case.

5.3 Results of the MCLNN's technique

The following are the results when the variable epochs is given the values [25, 50, 100]:

- Epochs = 25

Test accuracy	89.46 %
Test loss	0.349775
Percentage of correct predictions	89.45 %

Table 7. Metrics for MCLNN with epochs=25.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	87.7 %	0	2.1 %	2.0 %	0.4 %	0.2 %	7.2 %	0	0.4 %	0
Class 1	0.2 %	97.6 %	0.1 %	1.4 %	0.4 %	0	0.1 %	0	0.2 %	0
Class 2	2.4 %	0	86.0 %	1.0 %	6.0 %	0	4.4 %	0	0.2 %	0
Class 3	1.7 %	0.7 %	1.6 %	88.9 %	4.4 %	0	2.1 %	0	0.6 %	0
Class 4	0.2 %	0.1 %	11.8 %	1.2 %	79.4 %	0	7.1 %	0	0.2 %	0
Class 5	0	0	0	0.1 %	0	97.0 %	0	2.1 %	0.1 %	0.7 %
Class 6	13.8 %	0	8.3 %	2.6 %	4.6 %	0	70.0 %	0	0.7 %	0
Class 7	0	0	0	0	0	0.6 %	0	97.3 %	0.1 %	2.0 %
Class 8	0.5 %	0	0.4 %	0.4 %	0.2 %	0.1 %	0.4 %	0.5 %	97.5 %	0
Class 9	0	0	0	0	0	1.1 %	0.1 %	4.1 %	0	94.7 %

Table 8. Confusion Matrix for MCLNN with epochs=25.

- Epochs = 50

Test accuracy	89.61 %
Test loss	0.599841
Percentage of correct predictions	89.60 %

Table 9. Metrics for MCLNN with epochs=50.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	80.0 %	0.1 %	2.6 %	3.4 %	0.3 %	0.2 %	12.7 %	0	0.6 %	0.1 %
Class 1	0.3 %	96.8 %	0.3 %	1.8 %	0.5 %	0	0.2 %	0	0.1 %	0
Class 2	1.3 %	0	88.2 %	1.8 %	3.9 %	0	4.7 %	0	0.1 %	0
Class 3	1.0 %	0.3 %	1.3 %	91.7 %	3.9 %	0	1.3 %	0	0.5 %	0
Class 4	0.1 %	0	16.1 %	2.6 %	77.9 %	0.1 %	2.9 %	0	0.3 %	0
Class 5	0	0	0.1 %	0	0	93.5 %	0	3.0 %	0.2 %	3.2 %
Class 6	9.5 %	0.1 %	9.7 %	3.6 %	5.9 %	0	70.3 %	0	0.9 %	0
Class 7	0	0	0	0	0	0.7 %	0	96.3 %	0.1 %	2.9 %
Class 8	0.4 %	0	0.8 %	0.8 %	0.1 %	0.3 %	0.2 %	0.4 %	96.9 %	0.1 %
Class 9	0	0	0	0	0	0.2 %	0.1 %	2.0 %	0	97.7 %

Table 10. Confusion Matrix for MCLNN with epochs=50.

- Epochs = 100

Test accuracy	89.77 %
Test loss	0.793967
Percentage of correct predictions	89.77 %

Table 11. Metrics for MCLNN with epochs=100.

and **Confusion Matrix** is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	86.3 %	0.1 %	1.8 %	1.7 %	0.7 %	0.1 %	8.7 %	0.1 %	0.5 %	0
Class 1	0.1 %	98.3 %	0	0.9 %	0.2 %	0	0.3 %	0	0.1 %	0.1 %
Class 2	2.2 %	0.2 %	82.2 %	1.1 %	9.1 %	0	5.2 %	0	0	0
Class 3	1.2 %	0.4 %	0.6 %	90.4 %	4.5 %	0	2.3 %	0	0.6 %	0
Class 4	0.1 %	0	7.3 %	2.1 %	87.4 %	0	2.9 %	0	0.2 %	0
Class 5	0.1 %	0.1 %	0	0.1 %	0	96.6 %	0	1.6 %	0.2 %	1.3 %
Class 6	12.8 %	0.2 %	7.6 %	3.0 %	11.4 %	0	64.4 %	0	0.6 %	0
Class 7	0.1 %	0	0	0	0	1.5 %	0	97.4 %	0	1.0 %
Class 8	0.6 %	0	0.2 %	0.4%	0.5 %	0.1 %	0.4 %	0.2 %	97.6 %	0
Class 9	0.1 %	0	0	0	0	1.2 %	0.1 %	4.6 %	0	94.0 %

Table 12. Confusion Matrix for MCLNN with epochs=100.

We can see that as the number of epochs rises, the accuracy increases as well. This means that our Masked Conditional Neural Network iterates over the whole training dataset, providing better and more accurate predictions each time. It's also worth noting that when the number of epochs grows, the loss follows the same pattern. It's probable that overfitting is to blame here again. As we have already said, overfitting is a term used in data science to describe when a statistical model fits its training data perfectly. When this happens, the algorithm is unable to perform accurately against unknown data, negating the goal of the method.

As we have already done in previous cases, in MCLNN – Rectangle Mask Matrix, our extension, we will increase the variable of epochs and we will notice the values of the interesting metrics (accuracy, loss, percentage of correct predictions and Confusion matrix). So:

- Epochs = 25

Test accuracy	89.09%
Test loss	0.380718
Percentage of correct predictions	89.09 %

Table 13. Metrics for MCLNN – Rectangle Mask Matrix with epochs=25.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	86.5 %	0.4 %	2.2 %	2.1 %	0.5 %	0.1 %	7.2 %	0	1.0 %	0
Class 1	0	98.8 %	0	0.6 %	0.3 %	0	0.2 %	0	0.1 %	0
Class 2	1.7 %	0.1 %	86.3 %	1.3 %	7.7 %	0	2.9 %	0	0	0
Class 3	2.0 %	1.2 %	1.1 %	90.9 %	3.4 %	0	1.1 %	0	0.3 %	0
Class 4	0.1 %	0.1 %	9.7 %	2.8 %	85.3 %	0	1.8 %	0	0.2 %	0
Class 5	0	0	0	0.1 %	0	96.9 %	0	2.2 %	0	0.8 %
Class 6	13.7 %	0.2 %	11.9 %	3.3 %	12.0 %	0	57.8 %	0	1.1 %	0
Class 7	0	0	0	0	0	1.1 %	0	97.6 %	0	1.3 %
Class 8	0.4 %	0.1 %	0.2 %	0.5 %	0.5 %	0.3 %	0.8 %	0.4 %	96.8 %	0
Class 9	0	0	0	0	0	1.2 %	0.1 %	4.7 %	0	94.0 %

Table 14. Confusion Matrix for MCLNN – Rectangle Mask Matrix with epochs=25.

- Epochs = 50

Test accuracy	89.31%
Test loss	0.531949
Percentage of correct predictions	89.31 %

Table 15. Metrics for MCLNN – Rectangle Mask Matrix with epochs=50.

and Confusion Matrix is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	86.8 %	0.1 %	1.9 %	1.5 %	0.6 %	0.2 %	7.5 %	0	1.4 %	0
Class 1	0.4 %	97.9 %	0.1 %	1.0 %	0.4 %	0	0.1 %	0	0.1 %	0
Class 2	2.1 %	0.1 %	78.7 %	1.3 %	13.1 %	0	3.9 %	0	0.8 %	0
Class 3	1.7 %	0.4 %	1.3 %	91.3 %	3.4 %	0	1.5 %	0	0.4 %	0
Class 4	0	0.1 %	3.7 %	2.9 %	89.4 %	0	3.2 %	0	0.7 %	0
Class 5	0	0	0.1 %	0.1 %	0	95.8 %	0	2.7 %	0.1 %	1.2 %
Class 6	14.2 %	0.2 %	7.4 %	2.9 %	10.9 %	0	62.6 %	0	1.8 %	0
Class 7	0	0	0	0	0	0.5 %	0	98.3 %	0	1.2 %
Class 8	0.4 %	0	0.4 %	0.2 %	0.3 %	0.1 %	0.1 %	0.4 %	98.1 %	0
Class 9	0	0	0	0	0	0.8 %	0.1 %	4.9 %	0	94.2 %

Table 16. Confusion Matrix for MCLNN – Rectangle Mask Matrix with epochs=50.

- Epochs = 100

Test accuracy	89.37 %
Test loss	0.763854
Percentage of correct predictions	89.37 %

Table 17. Metrics for MCLNN – Rectangle Mask Matrix with epochs=100.

and **Confusion Matrix** is:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	87.0 %	0.1 %	2.0 %	0.9 %	0.3 %	0.2 %	9.3 %	0	0.2 %	0
Class 1	0.5 %	98.3 %	0	0.5 %	0.4 %	0	0.3 %	0	0	0
Class 2	2.1 %	0.1 %	87.1 %	0.5 %	4.0 %	0.1 %	6.0 %	0	0.1 %	0
Class 3	3.8 %	1.1 %	1.9 %	85.1 %	4.1 %	0	3.6 %	0	0.4 %	0
Class 4	0.3 %	0.3 %	127%	1.8 %	79.3 %	0	5.4 %	0	0.2 %	0
Class 5	0	0	0	0	0	96.9 %	0	1.8 %	0.3 %	1.0 %
Class 6	14.9 %	0.1 %	9.5 %	1.9 %	6.1 %	0	67.0 %	0.1 %	0.4 %	0
Class 7	0	0	0	0	0	0.6 %	0	97.2 %	0	2.2 %
Class 8	0.9 %	0	0.6 %	0.4 %	0.1 %	0.1 %	0.2 %	0.4 %	97.3 %	0
Class 9	0.1 %	0	0	0	0	0.8 %	0.1 %	2.7 %	0.1 %	96.2 %

Table 18. Confusion Matrix for MCLNN – Rectangle Mask Matrix with epochs=100.

The accuracy and loss's increase keeps pace with the rise in epochs – something happens because of the reasons we have broached before.

In all Confusion Matrices, we can clearly observe that the highest percentage of correct predictions in each class has gathered in the main diagonal, as it should be, while there are little wrong predictions for each class scattered in other classes. The highest value of accuracy for all experiments is **89.77 %** , which can be detected in case of Masked Conditional Neural Networks with the parameter of epochs equal to 100. At CLNNs the accuracy is 88.58 % on average while at MCLNNs and MCLNNs – Rectangle Mask Matrix is above 89 %, and on average 89.435 %. In total, our approach is very effective as the accuracy fluctuates between 88 – 90 % in all cases and all experiments.

Chapter 6. Conclusions and Discussion

In conclusion, MCLNN retains its generalization, allowing to be used to any other multi-dimensional temporal representation. MCLNN will be used as a stand-alone feature extractor for other pattern analysis tasks with the help of deeper architectures. Future work will include optimizing the mask patterns, considering different combinations of the order across the layers, and using the MCLNN as a stand-alone feature extractor for other pattern analysis tasks with the help of deeper architectures. We'll also investigate using the MCLNN on signals with a temporal component, such as EEG and EMG. Merging the MCLNN to extract the local feature with the long-term dependency collected by an LSTM, similar to a CRNN, will be investigated.

As happened with the Rectangle Mask Matrix, there will certainly exist other variations about the schema of Mask Matrix, which will probably lead to better results. So, the form of Mask and its functionality consist crucial parameters for further exploration.

Yet another factor that is open for study and exploration for next decades is the application of a different dataset on both Conditional Neural Networks and Masked Conditional Neural Networks, solving the image classification problem again. It will be a good opportunity to use other Image Classification Datasets such as Architectural Heritage Elements Dataset and Image Classification: People and Food Dataset. Furthermore, it will be attainable to turn to other Sciences, such as Medical and Agriculture, as they have flourished in recent years. So, some of them could be the Recursion Cellular

Image Classification Dataset, the CoastSat Image Classification Dataset, and the Intel Image Classification Dataset.

Last but not least, it is important to say that except for images there are other types of data available in order to use in Neural Networks' models analyzed before. There is a large spread of multidimensional data in recent years, a fact that is justified by the age we are living. So, multidimensional on their whole or just signals for more simplicity consist a significant chance for further analysis in Deep Learning generally.

In closing, no matter what the schemas of data or the techniques will be, the effort and the analysis should be in a mood for further education, knowledge, evolution, and development in all sections.

BIBLIOGRAPHY

- F. Medhat, D. Chesmore, and J. Robinson, **Masked Conditional Neural Networks for Environmental Sound Classification**, Cornell University, USA, 27 April 2019

- **Keras API reference** Website

- **IBM Cloud Learn Hub** Website

- **Machine Learning Mastery** Website

- G. Singh Sohi, **Conditional Neural Networks**, GOOD AUDIENCE, 5 July 2018

- S. Saha, **A Comprehensive Guide to Convolutional Neural Networks,** Towards data science, 15 December 2018

- Infolks Pvt Ltd**, Recurrent Neural Network and Long-Term Dependencies**, INFOLKS, 14 July 2019

- S. Swain, **Understanding Sequential Vs Functional API in Keras,** Analytics Vidha, 13 July 2021

- F. Medhat, D. Chesmore, and J. Robinson, **Masked Conditional Neural Networks for sound classification,** Applied Soft Computing, Vol 90, May 2020

- T. Gupta, Deep Learning: Feedforward Neural Network, Towards data science, 5 January 2017

- E. Kavlakoglu, AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference? , IBM, 27 May 2020

- J. Brownlee, Deep Learning CNN for Fashion-MNIST Clothing Classification, Deep Learning for Computer Vision, 10 May 2019

- Zalando Research, Fashion MNIST, Kaggle, Version 4, 2018

- E. Burns, machine learning, Techtarget, March 2021

- P. Vadapalli, Decision Tree in AI: Introduction, Types & Creation, upGrad, 3 September 2020

- MathWorks.com, What Is Deep Learning?