Master in Innovation and Research in Informatics

Designing Mixture of Deep Experts

Sai Krishna Kalyan

July 07/03/2017

Supervision : Christian Gagné
(Université Laval)

Lluís A. Belanche
(Polytechnic University of Catalonia)

Location : Université Laval, Canada

Abstract:

Mixture of Experts (MoE) is a classical architecture for ensembles where each member is specialised in a given part of the input space or its expertise area. Working in this manner, we aim to specialise the experts on smaller problems, solving the original problem through some type of divide and conquer approach.

The goal of our research is to initially reproduce the work done by Collobert et al[1] , 2002 followed by extending this work by using neural networks as experts on different datasets. Specialised representations will be learned over different aspects of the problem, and the results of the different members will be merged according to their specific expertise. This expertise can then be learned itself by a given network acting as a gating function.

MOE architecture composed on N expert networks. These experts are combined via a gating network, which partition the input space accordingly. It is based on divide and conquer strategy supervised by a gating network. Using a specialised cost function the experts specialise in their sub-space. Using the discriminative power of experts is much better than simply clustering. The gating network needs to needs to learn how to assign examples to different specialists.

Such models show promise for building larger networks that are still cheap to compute at test time, and more parallelizable at training time. We were able to reproduce the work by the author and implemented a multi-class gater to classify images.

We know that Neural Networks perform the best with lots of data. However, some of our experiments require us to divide the dataset and train multiple Neural Networks. We observe that in data deprived condition our MoE are almost on par and compete with ensembles trained on complete data.

Keywords : Machine Learning, Multi Layer Perceptrons, Mixture of Experts, Support Vector Machines, Divide and Conquer, Stochastic Gradient Descent, Optimization

Contents

1 Hosting institution

1.1 Université Laval

The presented work has been developed during an internship at **Université Laval**. Université Laval is a French-language, public research university in Quebec City, Quebec, Canada. The University was founded by royal charter issued by Queen Victoria in 1852, with roots in the founding of the Séminaire de Québec in 1663 by Francois de Montmorency-Laval, making it the oldest centre of higher education in Canada and the first North American institution to offer higher education in French.

Université Laval contributes to the development of society by educating qualified and responsible individuals who become agents of change and by seeking and sharing knowledge in a stimulating environment of research and creation. It is a major player in the Canadian research scene, Université Laval recent development plan flagged up in particular the overarching areas of culture, innovative materials, healthcare, education and the environment.

1.2 Unite mixte de recherche en sciences urbaines

UMRsu is a unique research and innovation network bringing together players from industrial, government, and the academic community for the purpose of developing smart and sustainable cities. The benefits from UMRsu research and innovation activities are estimated at 19.2 million and that does not include the some 3 million dollars in funding already secured to create the joint research unit and get it up and running.

By collecting, integrating, and analyzing data, the Joint Research Unit in urban sciences (UMRsu) explores the workings of the city in all its facets, for the benefit of its citizens. A living, city-wide laboratory, the UMRsu is headquartered at the heart of the Québec Metro High Tech Park. With 3,000 square feet of space, the UMRsu features a full-blown operations room equipped with interactive technology, a server room, and innovative software tools.

2 Acknowledgement

I would like to thank my supervisor Christian Gagné who first gave me the opportunity to do research internship at Université Laval. He provided me with the perfect balance of guidance and freedom. I would also like thank my co-supervisor Lluís A. Belanche for his valuable advice throughout the period of my research internship and MITACS for funding my research work.

I am grateful to all the professors from Lumière Université Lyon 2 (ULY2) and the Polytechnic University of Catalonia (UPC) for sharing their knowledge and time.

Also, special thanks to Audrey Durand for our insightful discussions on my thesis. I would also like to thank Mahdieh Abbasi, Marc-André Gardner and Yannick Hold-Geoffroy for their technical feedback.

Finally, I would like to thank my family who stood by me during all this time and friends for their love and support.

3 Introduction

Machine Learning is the study of algorithms which can learn from data. Machine Learning techniques are relevant to several areas such as data mining, computer vision and speech recognition, which are dealing with an increasing amount of data. One of the ways to improve the machine learning technique is by using ensemble methods. These methods use multiple learning algorithms to obtain better predictive performance.

We address only classification problems in our work. This thesis aims to address how to train specialised experts using Neural Networks(NN) with a gater. The experts can be any machine learning algorithm. The underlying idea of Mixture of Expert (MoE) is that a complex function can be decomposed to several simpler function where each expert learns on a different input space. We experiment changing of expert from Support Vector Machine (SVMs) to a NN / Convolutional Neural Network (CNN). Applications using this approach can be found in many domains like Speech Recognition[2], Image Classification[3] for example.

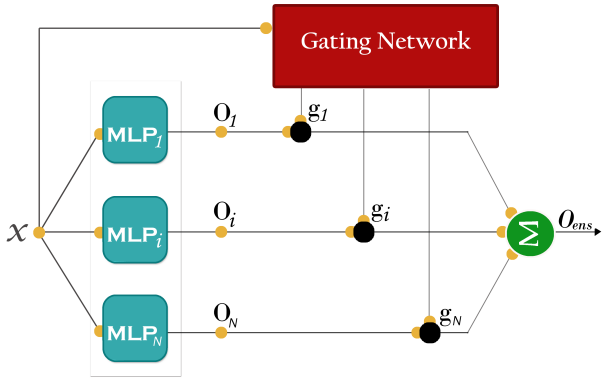

Figure 3.1: Simple expert network with N experts that specialize in a subset of data.

This thesis aims to be self explanatory assuming that the reader has some knowledge in machine learning. Thus in this chapter we formally define the framework and introduce the concept of MoE and also discuss what were the main contributions of this thesis. In Chapter 4 we will define MoE formally and will describe it in detail. This chapter will give us strong theoretical foundations for our experiments. We will introduce and explain common algorithms used in our experiments in Chapter 5. Chapter 6 deals with the technical details required to setup our experimental framework. Chapter 7 exposes the results that we obtained from our experiments. Finally in Chapter 8 we conclude with conclusions and describe our future work. In this chapter we discuss our findings and discusses shortcomings and advantages.

3.1 Fundamentals

In Machine Learning, **classification** is a general process related to categorisation. Classification is considered as an instance of supervised learning. An example would be assigning a given email into $spam$ or $non-spam$. Classification is an example of pattern recognition.

Hyperparameter Optimization is the problem of choosing a set of hyper-parameters for a learning algorithm. The goal of optimizing is a measure of the algorithms performance on an independent data set. Hyperparameter Optimization ensure the model does not overfit its data

by tuning. In our experiment we use grid search, which is a traditional way of manually defining a specified subset of hyper parameter space of the learning algorithm.

Ensembling is a general term for combining many classifiers by averaging or voting for example. It is a form of meta learning, it focuses on how to merge results of arbitrary underlying classifiers. Generally, ensembles of classifiers perform better than single classifiers. Consider having multiple NN models. Ensemble averaging creates a group of networks, each with low bias and high variance, then combines them to a new network with (hopefully) low bias and low variance, according the well known bias-variance tradeoff. Names of ensemble techniques include bagging, boosting, model averaging, and MoE. An example of an ensemble technique is the use of (plural) random forests over (singular) decision trees.

Error-Correcting Output Codes (ECOC) is an ensemble method designed for multi-class classification problem. For example, in digit recognition task, we need to map each hand written digit to one of $k = 10$ classes. Some algorithms, such as decision tree, naive bayes and neural network, can handle multi-class problem directly. ECOC is a meta method which combines many binary classifiers in order to solve the multi-class problem. In this approach we want to maximise the Hamming distance between each meta target. The main advantage of this method is robustness to individual classifiers error.

3.2 Contributions of this thesis

We propose a new approach to MoE by replacing the experts from SVM to CNNs. We also introduce prime network, whose mainly responsible is representation learning, this is very useful when dealing with high dimensionality data like images. Finally we propose a new subset of labels approach where we split data based on labels instead of observations.

4 Mixture of Experts

MoE is one of the most interesting ensembling methods. It is mainly based on divide and conquer principle where the problem space is divided between experts. This idea of MoE is quite old, It was first introduced to machine learning research community by Jacobs et al.[4], 1991. Its been more than 25 years and has been used for classification, regression with applications in healthcare, finance, pattern recognition etc. In the past 25 years, there has been several statistical and experimental analysis of MoE and several studies have been published [5]. We present some related work below.

Hierarchical mixtures of experts and the Expectation Maximazation algorithm was presented by Jacobs et al[6]. Where the authors presented a tree structured architecture for supervised learning. The learning was treated as Expectation Maximisation (EM) problem. For mixture of experts architectures, the EM algorithm decouples the estimation process in a manner that fits well with the modular structure of the architecture.

In Divide-and-conquer[7] approach, the authors divide the training set using an unsupervised algorithm to cluster the data. Then we train each expert SVM on each subset corresponding to each cluster. In this approach there is no notion to reassign data after the gater network has been trained.

Bayesian Committee Machine[8] is a technique to partition the data into many subsets, we train SVMs in individual subsets and then use a specific combination scheme based on the covariance of test data to combine the predictions. This method scales linearly to training data however cannot operate on a single test example. Like in the previous case it assigns examples randomly to the experts.

There is another paper where authors extend the Mixture of Experts to a stacked model with multiple sets of gating and experts. This exponentially increase the number of effective experts by associating each input with a combination of experts at each layer and yet maintain a modest model size. This Mixture of Expert learnt to develop location dependent experts at the first layer and class specific experts at the second layer. This work was known as Learning Factored Representations in a deep mixture of experts[9].

Sparsely-Gated Mixture-of-Experts Layer[10] proposed an approach were conditional computing could be used where parts of neural network are active on per example basis. This approach offers 1000x improvement in model capacity consisting of thousands of feed forward sub networks. Here the authors present an MOE architecture with 137 billion parameters applied convolutionally between stacked LSTM layers and achieve significantly better results than the state-of-the-art.

Distilling the Knowledge in a Neural Network was a presented by Hinton et al.[11] which is a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

More recent work by Abbasi et al.[12] include using ensemble of diverse specialists where speciality is defined according to the confusion matrix. In this work the author argues that an ensemble of specialists should be better able to identify and reject fooling instances, with a high entropy (i.e., disagreement) over the decisions in the presence of adversaries.

4.1 Divide and Conqure

The basic idea behind MoE is to divide the input space into several parts, to build a model for each part and then combine the local models to get a better global model. This approach is used to solve a complex problem by diving it into few simpler problems which have simpler solutions. The solutions to simpler problems are then combined to yield a solution to the complex problem. Typically divide and conquer problem have two modules. A module to divide the problem in to smaller problems and another module to decide how to combine the results of local problems. This principle of divide and conquer and is quite popular in several fields such as computer science and applied mathematics etc.

4.2 Loss Function

In this section we describe the loss function for the gater. The output of the mixture for a given input vector x is computed as below.

$$f(x) = h(\sum_{m=1}^{M} w_m(x) s_m(x)) \tag{1}$$

Here M is the number of experts in the mixture, $s_m(x)$ is the output of the m^{th} expert given input x, $w_m(x)$ is the weight for the m^{th} expert given by the gater module. h is the transfer function associated which could be hyperbolic tangent or softmax for classification tasks.

In multi-class classification setting our softmax classifier uses the cross entropy loss. The categorical cross-entropy loss is also known as the negative log likelihood. It is a popular loss function for categorisation problems and measures the similarity between two probability distributions, typically the true labels and the predicted labels. In an information theory point of view, the cross-entropy between true distribution p and estimated distribution q is defined as:

$$H(p,q) = -\sum_x p(x) * \log q(x) \tag{2}$$

Hence the softmax classifier minimises the cross-entropy between the estimated class probabilities and the true distribution. Cross entropy can also be written in terms of entropy and the Kullback-Leiber divergence as

$$H(p,q) = H(p) + D_{kl}(p\|q) \tag{3}$$

The entropy of the delta function p is zero, this is also equivalent to minimising the KL divergence between the two distributions. In other words the entropy objective wants the predicted distribution to have all of its mass on the correct answer.

4.3 Gater

A standard mixture of experts represents a soft decomposition of data into subsets, thus both the gater is trained on the whole data and the expert on the subset of data. The gater can be considered to be a simple neural network.

It has two main functions, the first one is to combine the input data with output of experts. This is achieved by a simple dot product. In this dot product input data is represented as the output of the neuron which is equal to number of experts. The second function is to learn the weight which will be used to re-distribute data based on the weights assigned to each observation. This loop continues until the validation error increases. Experts learn from local input subset while the gater learns from the combination of local expert subset outputs and global input data.

5 Algorithms

In this section we describe common classification algorithms we used in our experiments. We used SVMs for binary classification and multilayer perceptrons(MLPs) and Convolutional Neural Networks(CNN) for multi class classification problem.

5.1 Support Vector Machine

In 1992 Vapnik and coworkers proposed a supervised algorithm for classification that has since evolved into what are now known as Support Vector Machines)[13] (SVMs). The key innovations of this method was the explicit use of convex optimization, statistical learning theory, and kernel functions. SVMs are supervised learning models with associated learning algorithms that analyze

Algorithm 1 Training Mixture of Experts

Data: $[train, val, test] = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
Divide the data into K random disjoint subsets D_k of size near N/K.
$D = \{D_1, D_2, ...D_k\}$
Train each expert separately on D_k
Train gater by minimising equation 1
for l **do**
    $[w_1, w_2, ..., w_m] = MLP(x_i)$
    Sort the experts in descending order according to w_m
    if Expert has more than (N/K + c) examples **then**
        Assign this example to an expert with less than (N/K + c) examples
if Validation error increases **then**
    Terminate the algorithm
return $C = \{Expert_1, Expert_2, ..., Expert_N\}, Gater$;

data used for classification and regression analysis.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick. Support Vectors are the data points that lie closest to the decision surface. They are most difficult to classify and have a direct bearing on the optimum location of the decision surface. We can show that the optimal hyperplane stems from the function class with the lowest capacity (i.e CV dimension).

$$y = sign(\sum_{n=1}^{N} y_i \alpha_i K(x, x_i) + b) \tag{4}$$

SVMs are one of the best of the shelf supervised learning algorithm. Unique features of SVMs and Kernel Methods are that they are based on theoretical model of learning. They also come with theoretical guarantees about performance and do not suffer from curse of dimensionality.

5.2 MultiLayer Perceptron and Deep Learning

A MultiLayer Perceptron(MLP) is a feedforward artificial neural network. It maps the input data to appropriate output set. Each neuron activation function which is usually used for producing non linear mapping between the input and the output ranging between 0 to 1. Traditionally an MLP consists of two to three layers stacked on top of each other.

One remarkable property of neural networks, widely known as universal approximation property, roughly describes that an MLP can represent any function with more than on hidden layer. Each node in one layer is connected to every node in the following layer also sometimes referred to as fully connected network.

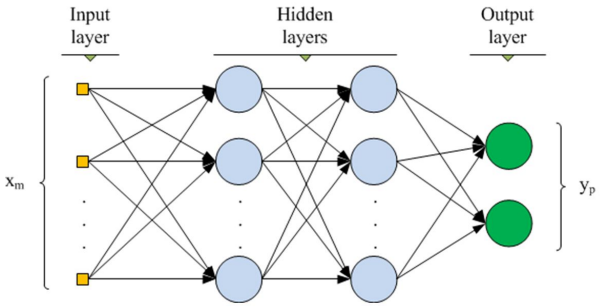

Figure 5.1: Simple MLP architecture

In the equation below we use the nomenclature of inputs being the first layer. The last layer (here the lth layer) is then given to an output layer on which an error function is minimised. Standard examples are softmax classifier and training with cross-entropy error, or a linear output layer with least square error.

$$z^2 = W^1 + b^1$$

$$a^2 = f(z^2)$$
$$z^3 = W^2 a^2 + b^2$$
$$a^3 = f(z^3)$$
$$\ldots$$
$$z^{l+1} = W^l a^l + b^l$$
$$y^{l+1} = f(z^{l+1})$$

Quite recently Deep Learning[14] methods have dramatically improved the state of the art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery. The deep refers to neural nets with more than one hidden layer.

The depth of the neural net allows it to construct a feature hierarchy of increasing abstraction, with each subsequent layer acting as a filter for more and more complex features that combine those of the previous layer. This feature hierarchy and the filters which model significance in the data, is created automatically when deep nets learn to reconstruct unsupervised data. Because they can work with unsupervised data, which constitutes the majority of data in the world, deep nets can become more accurate than traditional ML algorithms that are unable to handle unsupervised data.

5.3 Convolution Neural Network

CNN have been at the heart of deep learning. CNNs have been used as early as 1997. Now deep CNNs are considered to be state of the art in image classification tasks. We describe a simple architecture below.

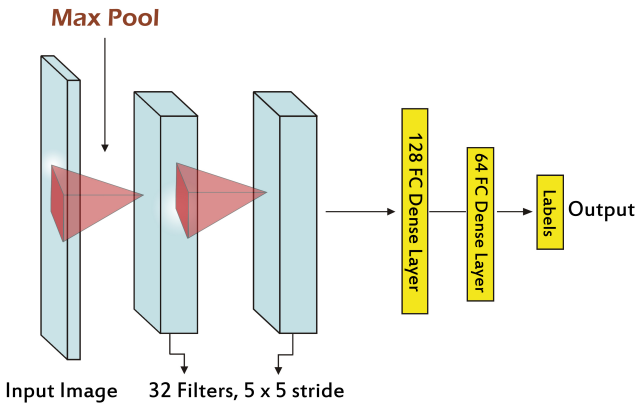

Figure 5.2: Convolutional Neural Network used in our experiments

Input is considered to hold the raw pixel value of the image. For example It could be 32 x 32 for an image in the CIFAR-10 database with three color channels Red, Blue and Green.

Convolutional layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input space. This may result in tensor of 32 x 32 x 12 dimensions, if we decided to use 12 filters.

ReLU layer will be apply an element wise activation function, such as $max(0, x)$ thresholding zero. This leaves the size of the volume unchanged.

Max Pool layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as 16 x 16 x 12.

Fully connected layer will compute the class scores, resulting in volume of size 1 x 1 x 10, where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the members in the previous volume.

6 Proposed Approach

The authors Collobert et al[1], demonstrated success using SVM experts. Some questions we wanted to answer are, does this approach work well with other type of experts?. In this section we describe our approach by using CNNs as experts and describe prime network and propose a new subset of labels approach. While dealing with multi classification problems our gating function had to be tweaked to accommodate dot product of tensors.

6.1 Deep Experts

We decide to use CNNs as experts instead of SVMs. We know that CNN are quite good at generalising images. We want to know if specialised representation could be learnt by individual CNN. We propose to establish a base line with a single CNN and compare it against the gated CNN.

We had to make choices like the type of architecture to use (We decided to use a modified version of Lenet) (ref), number of hidden units, learning rate, activation functions etc. We observed in our experiments that our NN and CNNs learn to specialise in a multi class classification setting.

Configuration of our network had two convolutional layer with 64 filter and a stride of 5 x 5 with max pooling between each convolutional layer, followed by three fully connected layers of size 128, 64 and 10. Each of these layer had dropout in the middle with *ReLU* activation.

6.2 Gater with Prime Network

When dealing with images we have high dimensional data. Before feeding this information to the gater we need to reduce the number of dimensions. We decided to train a prime network that we had the same architecture as that of the experts. We train this network on all the input observations and extract the output obtained from the $N-1$ layer of the neural network. This representation will have the same number of dimensions as that of the hidden units observed in this layer. This way we can easily reduce the number of dimensions of the input data and used this representation to train our gater. The configuration of our gater can be found in appendix.

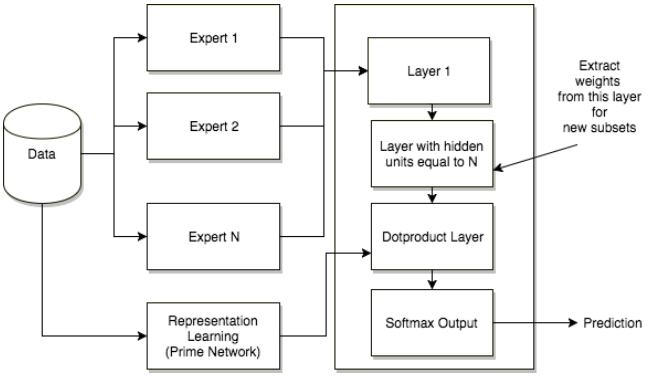

Figure 6.1: MoE with prime network for representation learning. We describe one iteration in the above diagram.

We begin by training a representation of our input data in lower dimension with the help of a prime network. In the first iteration we start by training our experts on random subsets. The output of this expert, prime network is fed as input to our gater. We save the required error metrics for this iteration. After the first iteration we use the weight obtained by our gating network (Which is equal to number of experts) to reassign our data and continue until we meet the stopping condition.

6.3 Gater with Subset of Labels

In this approach we split our data by labels instead of observations. This means that each expert would receive data that match a subset of labels. We decide to assign 5 labels to each expert. This assignment was not random as it would cause a lot of variance in our test error performance. We used a method inspired from ECOC described in section above to make sure that each expert got different labels overall (by maximise the Hamming distances between the pairs of label subsets). It is important to note that like in our earlier approach we had data reassignment after each iteration, we do not do this here.

7 Experimental Framework

All experiments in this thesis follow the same protocol. We divide the database into three subsets: a *training* set, a *validation* set and a *test* set. The training set is used for training the machine learning algorithm. The validation set is used for tuning the various hyper-parameters of the algorithm. The best hyper-parameters are chosen according to the classification error rate found on this subset. Finally the test is used to evaluate the generalisation performance between several algorithms. Our datasets were split using stratified sampling.

7.1 Tools

We used Keras[15] with Tensorflow[16] backend to conduct our experiments. TensorFlow is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation.

We also used scikit-learn[17] to re-use general purpose machine learning APIs like SVM and K-Nearest Neighbours. This project was started in 2007 as a Google Summer of Code project by David Cournapeau. Later that year, Matthieu Brucher started work on this project as part of his thesis.

7.2 Infrastructure

Our experiments were conducted on *Helios* super computing cluster. Most of our deep learning computations were done on Tesla K80 machines. Each compute node had 8 GPUs with 20 CPU cores and 128 GB RAM.

7.3 Datasets

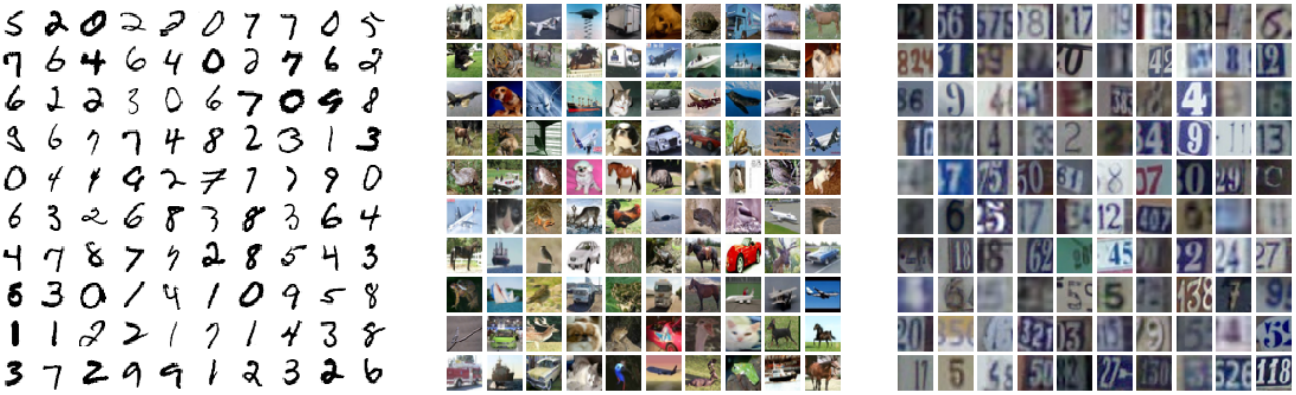

Figure 7.1: Sample images from MNIST, CIFAR10 and SVHN dataset

The Forest[1] database is currently the largest database available on the UCI machine learning repository. This database was used in the original paper on Mixture of Experts by Collobert

[1]https://archive.ics.uci.edu/ml/datasets/covertype

et al[1], 2002 to conduct their experiments and benchmark their results. This dataset includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. These areas represent forests with minimal human-caused disturbances, so that existing forest cover types are more a result of ecological processes rather than forest management practices. This database contains more than 500,000 observations and each observation in the database corresponds to a 30 x 30 square meter cell of forest, represented by 54 dimensional input vector with 10 continuous variables and 44 binary variables. The task is to determine the type of forest in each cell. This leads to a classification task with 7 classes. We split our dataset into training set with 100,000 examples, validation set with 50,000 examples and a test set with 10,000 examples.

The MNIST[2] database of handwritten digits[18], has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. We chose this database as its really good to get started with deep learning while spending minimal efforts on pre-processing. The MNIST training set consists of 28 x 28 black and white images and 10 class labels with values between 0 to 9.

The CIFAR-10[3] dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We choose to use this database as it is an established computer-vision dataset in deep learning. The label classes in this dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

The Street View House Numbers[4] (SVHN) is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST, but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. The SVHN data set contains 32 x 32 colour images with 10 classes.

Dataset	Traing Set	Validation Set	Testing Set
Forest	100k	10k	50k
MNIST	50k	10k	10k
CIFAR 10	40k	10k	10k
SVHN	63k	10k	26k

Table 1: This table describes our train, test and validation split of our datasets.

7.4 Pre-Processing

Based on the type of data we decide the pre-processing steps. We begin by transforming Forest dataset to binary classification problem followed by dividing each feature by the maximum found in the training set.

For images we one hot encode the the targets (Incase of SVHN dataset we had label classes from 1 to 10. We converted class 10 to 0). We also had to transform the input data to $float32$

[2]http://yann.lecun.com/exdb/mnist/
[3]https://www.cs.toronto.edu/~kriz/cifar.html
[4]http://ufldl.stanford.edu/housenumbers/

data type. Other per-processing steps were subtracting the image with its mean and division by a maximum pixel value of 255.

7.5 Evaluation

We use *Error* to evaluate how well our model did while classifying examples. For our initial experiments on *Forest* dataset we evaluated computational time take in sequential and parallel setting. In supervised learning applications in machine learning and statistical learning theory, generalization error (also known as the out-of-sample error) is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

7.6 Network Tuning

Based on our experiments conducted, I felt that choosing a good learning rate was the most critical tuning parameter. Using the default parameter could sometimes hurt the training process. Each of our dataset based on the type of NN architecture being used might need to be trained to different learning rate.

To make sure that our CNN does not get stuck in a saddle point we explicitly train for at least 100 epochs. After 100 epochs we decided to stop based on increase in validation loss. This decision was based when the mean of last ten epochs is greater than the last fifty epochs.

We decided to stick with Adam optimizer for all our experiments as it generally tended to be more stable. There were some experiments that gave us better results with RMSProp and Stochastic Gradient Descent optimiser however generally Adam optimiser worked well.

Usually visualising the loss curve was really important to see how our network was learning. Just by looking at the loss curve we can observe if our network is learning well or if there is something wrong with the loss function or even the architecture. Using tools like tensorboard we could see if our loss was generally decreasing and give us a good intuition to decide on what learning rate to use. In our experiments we observed that for MNIST and SVHN dataset the best learning rate to be 0.0001 and for CIFAR 10 dataset the best learning rate was 0.001.

For our initial experiments we did our best to follow exactly the configuration settings mentioned by Collobert et al.[1] like *tanh* activation MSE loss function etc. In our latter experiments we used $ReLU$ activation and categorical cross entropy loss.

7.7 Gater Tuning

Tuning gater is as important as tuning the NN. We need to decide the learning rate, Number of hidden layer, hidden units, activation function etc for the gater. Some times overlooked, its important to decide if the bias term should be included or not. In our experiments we observed an improvement when we remove bias from one of the layers (The layer where number of hidden units was equal to number of experts). We also saw improved performance when we increase the number of hidden layer from one to two for SVHN dataset.

We observed that once the best data subsets were learnt, the gating function had an bad loss curve. This is expected as the gater cannot be expected to learn better weights to partition the

data, hence we saw a drop in performance.

8 Experiments

This section describes the experiments conducted on various datasets. We begin by designing a MoE for binary classification followed by an implementation of a multi-class MoE for multi-class classification. We follow the steps described in **Algorithm 1**. First we decide the number of experts and divide the data into that many subsets. We train these subsets with experts and feed our input data and output of experts into the gater. We record overall error observed. We redistribute these subsets based on the weights obtained by the gater and continue this process until convergence.

8.1 Experts for Binary Classification

SVMs are very good for many classification problems. However they suffer from quadratic runtime complexity with respect to number of training examples. Hence, in the era of big data its difficult to try and solve real-life problems with more than hundred thousand examples with SVM. The paper by Collobert et al.[1] proposes a new mixture of SVMs that can be easily implemented in parallel and where each SVM is trained on a small subset of the whole data. We observed linear improvement in time complexity and a significant improvement in the generalisation process. In this sections below, we will describe how we went about empirically to reproduce results obtained by the original authors.

8.1.1 Reproducing Mixture of Experts

After doing the pre-processing steps described in the sections above we used grid-search to search for the parameters C and *gamma* with radial basis function (RBF) kernel.

The best values for C and *gamma* were obtained by grid-search over different validation sets. We found that the value of C was close to 10 and *gamma* around 6. In this grid-search we used 3 fold cross validation. We searched over 30000 input parameters over logarithmic scale. The search space of C ranged from 10^4 to 10^6 and for *gamma* from 10^{-5} to 10^6.

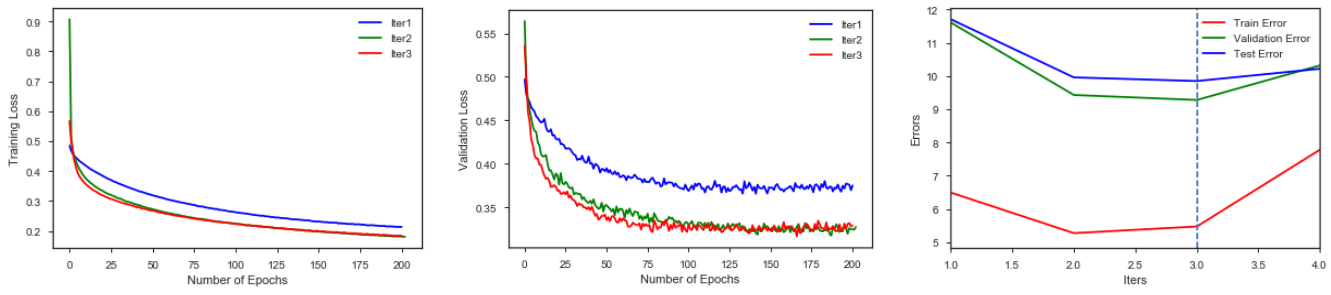

Figure 8.1: The plot on the right and center show training and validation loss observed in one of our experiments. Plot on the left shows us the overall result of the experiment. We observe a general trend that as our errors decrease as number of iterations increase.

This gating network uses mean squared error loss. All layers used the default sigmoid activation except the last layer which has tanh activation. We choose *Adam* optimiser with slightly high

learning rate 0.01. This gating network is responsible for the expertise level of SVM to classify an input sample. Iterations stop when validation error increases.

Experiment	Train Error	Test Error	Time (Sequence)	Time (Parallel)
Single MLP	11.72	14.43	13	
Single SVM	9.85	11.50	25	
Uniform SVM	16.98	17.65	15	10
Gated SVM	4.94	9.54	89	43
Gated MLP	17.27	17.66	137	

Table 2: Comparison of performances of Single MLP, SVM, SVM Mixture with equal weight to each expert and mixture of SVM. Result for fixed training set of 100,000 examples, validation set of 10,000 examples and test set of 50,000 examples. The selected model had 50 experts and 150 hidden units. Time taken is in minutes. Gated MLP model contains 10 experts.

We stop iterating as soon as the validation error goes up as shown in Plot 8.1 (at iteration 3) and break the loop. These results are encouraging and we could train several SVM like models on very large datasets in a reasonable amount of time.

8.1.2 Summary

The gated SVM mixture outperforms various divide and conquer methods as show in Table 2. This method is better than a single MLP or a single SVM. Since our dataset was small we can observe that our gated MLP mixture with 10 experts does not perform as well as the SVM experts. We also observe a clear computational advantage when our SVM experts are trained in parallel over multiple CPUs. In the sections below we modify our MoE approach by using CNNs as our experts instead of SVMs.

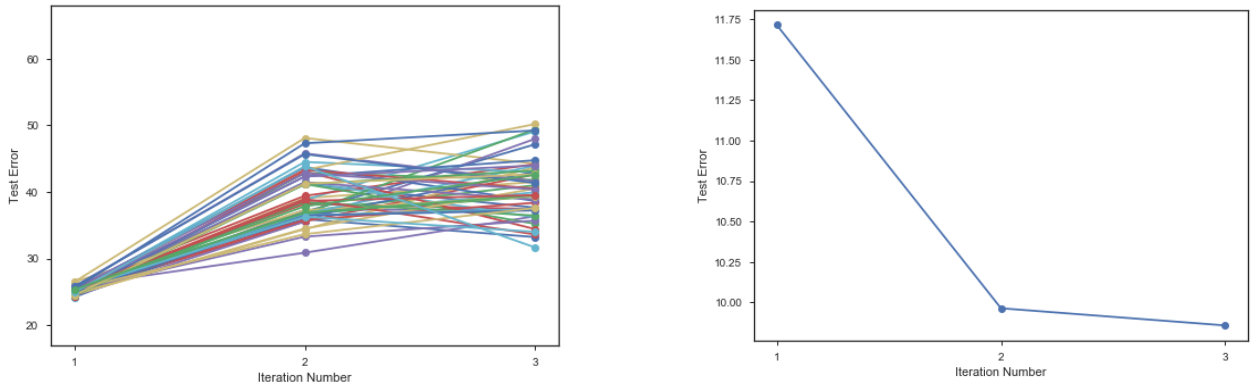

Figure 8.2: Each color represents an individual expert. Individually the experts become worse as the iterations increase, while the overall performance of the gater increases. This is expected as they initially are given random samples and generalise better and eventually they expertise and generalise worse. The other figure on the left shows that the test error decresases overall for the MOE as the number of Iterations increase.

We also observe a big drop in the test error after the first iteration and later the test error tends to flatten and converge as the number of iterations increase. Convergence is usually expected within 5 iterations in our experiments. Its important to understand the properties of experts before changing them. For example we noticed that SVMs to extremely well when the size of data is small. However we cannot say the same for an MLP. In sections below we will see that having less data can really hurt MLPs performance.

8.2 Effect of Data on a Convolutional Neural Network

We wanted to observe the performance of training error on various dataset sizes. Here we can easily see in the plot as the dataset size increases the test error decreases. We took stratified samples to train our CNN on different sizes. More data is always better for a neural network. Hence its important to note that in our MoE experiments our experts only learn from a subset of data due to the divide and conquer approach and hence we observe a performance degradation. (Each dataset have $10,000$ less examples which were removed for validation set.)

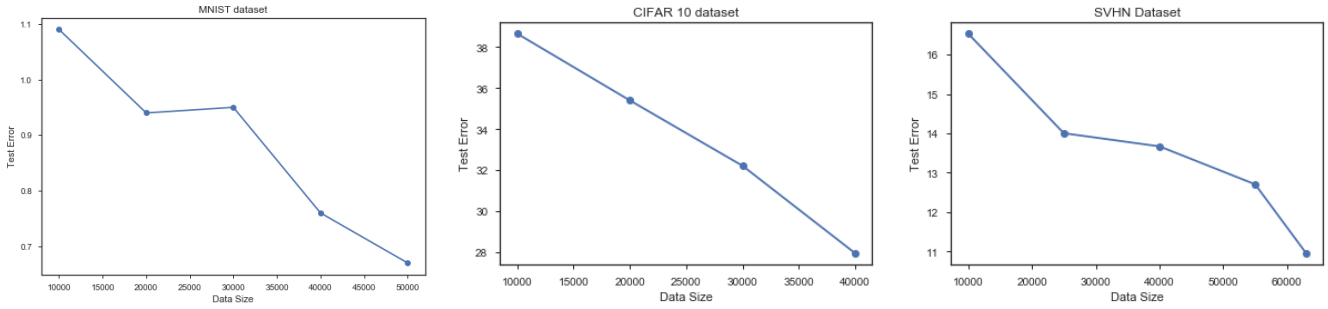

Figure 8.3: Effect of training error on our convolutional neural network on various datasets.

On MNIST data each would expect receives around 5000 observations (Considering we have 10 experts for all our experiments) with each expert having a test error of more than 3 percent. On CIFAR-10 dataset we have smaller dataset size and each of our expert would have a test error of around 45 percent and on $SVHN$ dataset an expert would have test error of around 19 percent.

8.3 Experts for Multiclass Classification

Multi-class classification is the problem of classifying instances into one of the many classes. Neural Networks and Multilayer perceptrons provide a natural extension to the multi-class classification problems. Since the dimensions of the input image is quite large we pass it through the prime network to reduce the input dimension. We pass the output again to a gater that has 2 layers with 50 hidden units each. Next layer has number of hidden units equal to number of experts (In our case the number of experts across all datasets is 10). This output is multiplied (dot product) with the output obtained from the experts. The output of this layer is again fed into a dense layer with number of hidden units equal to number of targets with softmax activation. Each layer is activated with $ReLU$ activation function except the last layer. Its important to note that we use a drop out of 0.5 to between each layer and we use categorical cross-entropy the loss function.

8.3.1 Results on Various Datasets

In this section we describe results obtained on various datasets across all experiments. To maintain consistency across various experiments we fixed the number of experts to 10 and used modified version on LeNet as described in Figure 3. All results seen below have been repeated 5 times to reduce variance and noise in our results. We consider Uniform CNN Split (Ensemble with each CNN receiving 1/10 of the data) as our baseline and Uniform CNN (Ensemble with each CNN receiving all the data) as our gold standard.

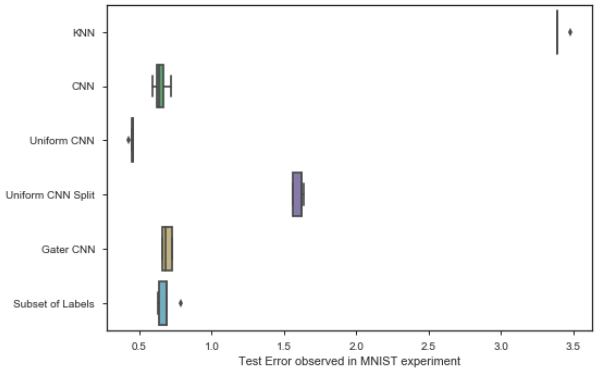

Figure 8.4: Test Errors observed MNIST dataset.

Experiment	Train Error	Test Error
K-NN	1.543	3.408
CNN	0.006	0.647
Uniform CNN	0.002	0.451
Uniform CNN Split	1.735	1.599
Gated CNN	0.057	0.69
Subset of Labels	0.022	0.68

Table 3: In this table we show the results obtained on MNIST dataset. We use 3 Nearest Neighbours as our baseline.

On MNIST dataset we know that K-Nearest Neighbours give us a good baseline to start with, less that 4 percent test error is observed. Experiments show that Gater CNN and Subset of labels have similar performance and do better than the baseline with a test error of less than 0.07 percent.

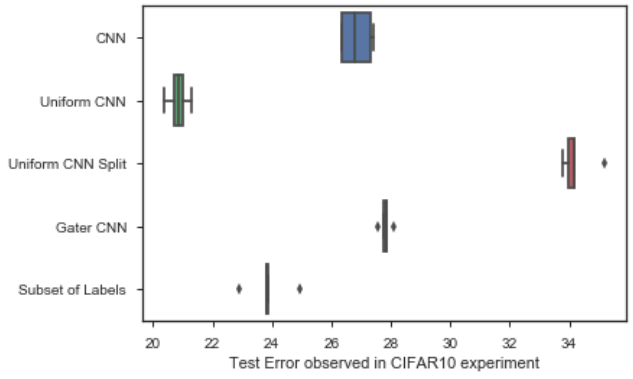

Figure 8.5: Test Errors observed CIFAR dataset

Experiment	Train Error	Test Error
CNN	1.415	26.84
Uniform CNN	0.080	20.83
Uniform CNN Split	28.482	34.251
Gated CNN	7.055	27.827
Subset of Labels	0.055	23.865

Table 4: In this table we show the results obtained on CIFAR-10 dataset.

CIFAR 10 dataset contains only $50k$ training examples. After removing $10k$ observation for validation we reduce this number to $40k$. This means each expert gets around $4k$ observations to learn from, each expert in Gater CNN has a test error of more than 45 percent (This can be observed in Plot 8.3). It overcomes this limitation and with the Gater it reduces the test error to 28 percent. All our Gater CNN and Subset of labels do better than the baseline.

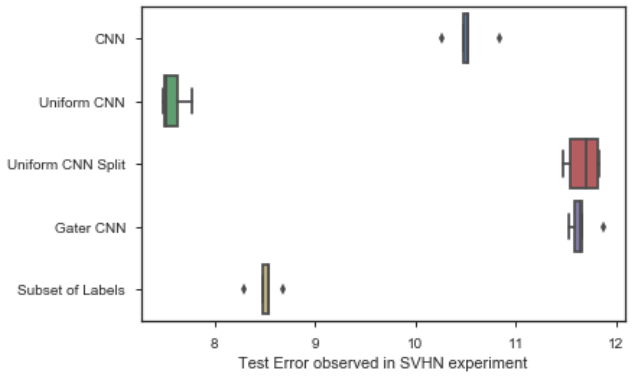

Figure 8.6: Test Errors observed SVHN dataset

Experiment	Train Error	Test Error
CNN	1.710	10.51
Uniform CNN	0.480	7.584
Uniform CNN Split	9.083	11.669
Gated CNN	3.351	11.652
Subset of Labels	0.393	8.504

Table 5: In this table we show the results obtained on SVHN dataset.

On SVHN dataset we observed that the performance of the Uniform CNN is comparable to subset of label. This is mostly because this dataset contains more than 60k images. This would allow the experts to learn from more data compared to all previous image datasets. Our Gater CNN and Subset of labels do better than the baseline. Gater CNN does not do very well, the main reason for this could be because of the input data is lot harder to classify. The expert needs to classify the number in the middle correctly.

We also wanted to check if our gater test error were robust and significant. We used Wilcoxon non-parametric test for our significance test. In Wilcoxon rank-sum test, the null hypothesis is that the two sets of measurements are drawn from the same distribution. The alternative hypothesis is that values in one sample are more likely to be larger than the values in the other sample. This is a non-parametric test and it does not require the assumption that of normal distributions. We made sure there was consistency in data across the gater experiments by using the same staring indices for each expert and use a p-value threshold of 0.05.

H_0 : Test error observed in CNN and Gater experiments are drawn from the same distribution.

H_1 : Test error observed in these experiments are not drawn from the same distribution.

The p-value observed for all our datasets was lower than 0.05 (p-value of 0.009) we can reject the null hypothesis and conclude that the test errors are draw from different distributions.

8.3.2 Summary

We already highlighted the need for more data to train experts. For MLP as experts we observed convergence within 3 iterations mostly. Also Plot 8.3 showed us there is almost a linear decrease in error as the number of training observations increase. Despite these issues we observe that our MoE and subset of labels have a comparable performance to uniform ensemble of CNN trained on complete data. Subset of labels does a better job than MoE in all the experiments because of the advantage of having more data.

9 Conclusions and Future Work

While conducting our experiments we observe an obvious thing, that having less data really hurts the neural network performance. In spite of this limitation we see that our MoE does almost as well as uniform averaging (Where each model receives complete data). Our experiments are easy to parallelise to gain computational performance improvement.

We would also like to use bigger datasets like the Imagenet [5] and observe its performance using this approach. This dataset would help experts better as the dataset size is lot larger and it would be interesting to see its performance in subset of labels. We could randomly assign several labels to each expert as this dataset has several categories.

Re-using the weights that we learnt from our prime network on our experts could also be an interesting idea that we would like to explore. These experts would converge faster as these weights have a better starting point compared to random initialisation.

Currently we are assigning the one observation to an expert based on weights obtained by the gater. We would like to observe MoE performance when we assign 2,3 or more experts the same observation. This would help the MOE to improve as it would have the advantage of learning from mode data (Incase of experts as MLP).

Subset of Labels would benefit from adding a dustbin class that contain half of the examples that do not belong to the class seen by the experts. So these experts would know how discriminate better as they have the knowledge of incorrect examples. We have some experimental results with data augmentation technique. However these experiments are not yet complete.

Some other experiments we would like to conduct would be to automatically decide the number of experts required based on heuristics like for example size of data or based on cross validation. We would like to explore deeper architectures observe their results.

[5] http://www.image-net.org/

ANNEXES

A First Appendix

CIFAR-10 has uniform class distribution. SVHN has more of ones and the class distribution is skewed to the right. MNIST dataset has unequal class distribution.

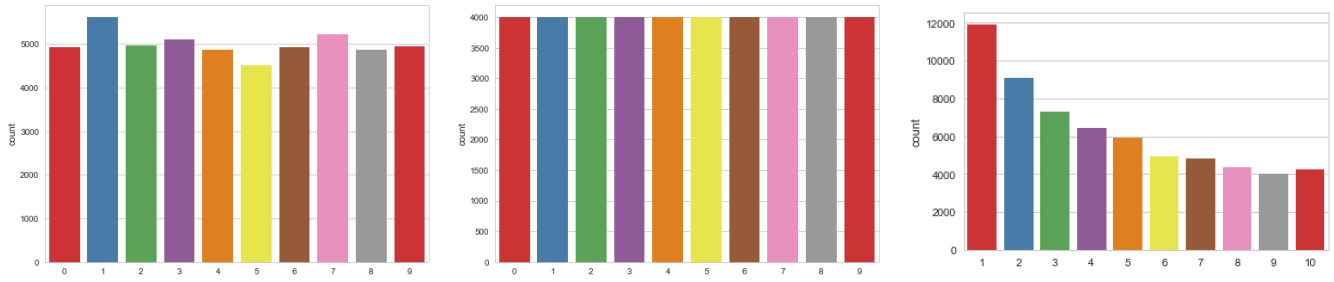

Figure A.1: This plot show the class ditribution for MNIST, CIFAR10 and SVHN dataset.

The biggest dataset we had was SVHN dataset followed by MNIST and CIFAR10.

Below we show results from a MoE on MNIST dataset. Here we see that this experiment had the best validation error of 0.719 at iteration 5.

Iteration	Train Error	Validation Error	Test Error
1	0.744	1.270	0.890
2	0.119	0.849	0.719
3	0.105	0.790	0.770
4	0.078	0.729	0.760
5 *	0.068	0.719	0.729
6	0.063	0.749	0.700

Table 6: Gater result on MNIST dataset. Here we see that training error decreases as the number of iterations increase. We stop when the iterating when the validation error increases.

B Second Appendix

In this section we show some NN architectures used in our experiments.

Forest MLP is the baseline MLPs used for the Forest experiments with *tanh* activation between its layers. We used *SGD* optimizer. Best performance was observed with high learning rate 0.1 and a momentum of 0.9.

Modified LeNet was used for all experiments with datasets with images. We used *ReLU* activation with two convolutional layer and two max-pooling layers. This was followed by three fully connected layers with softmax activation.

The **Gater** we used had *ReLU* activation for each of the layers with softmax activation for the last layer. The learning rate differed based on the type of dataset we were working on with adam optimizer and categorical crossentropy loss function.

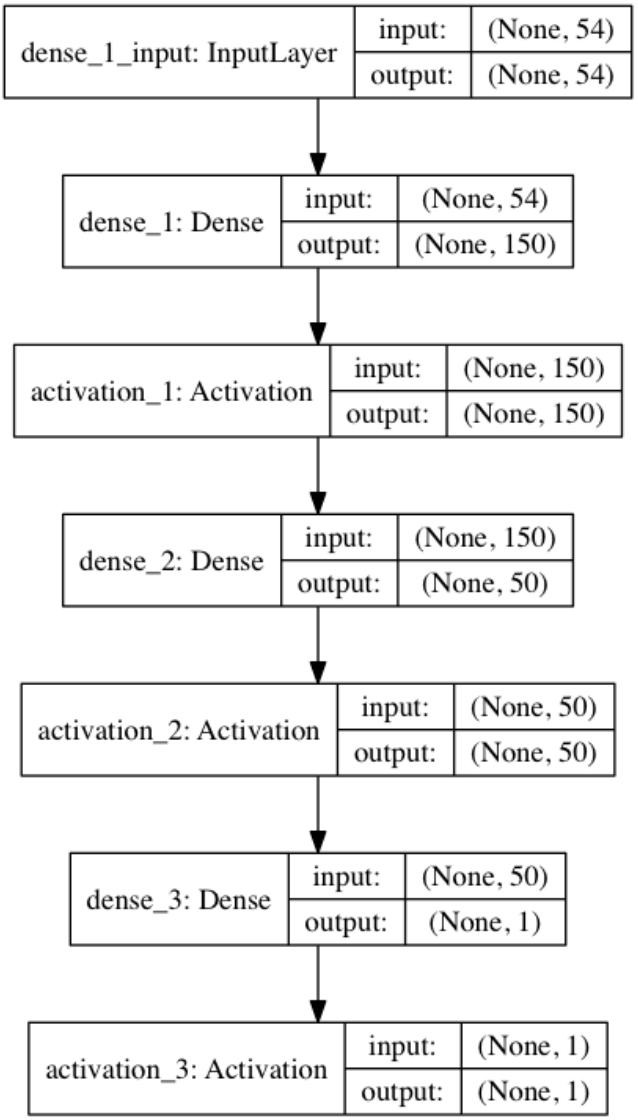

Figure B.1: MLP used in Forest dataset

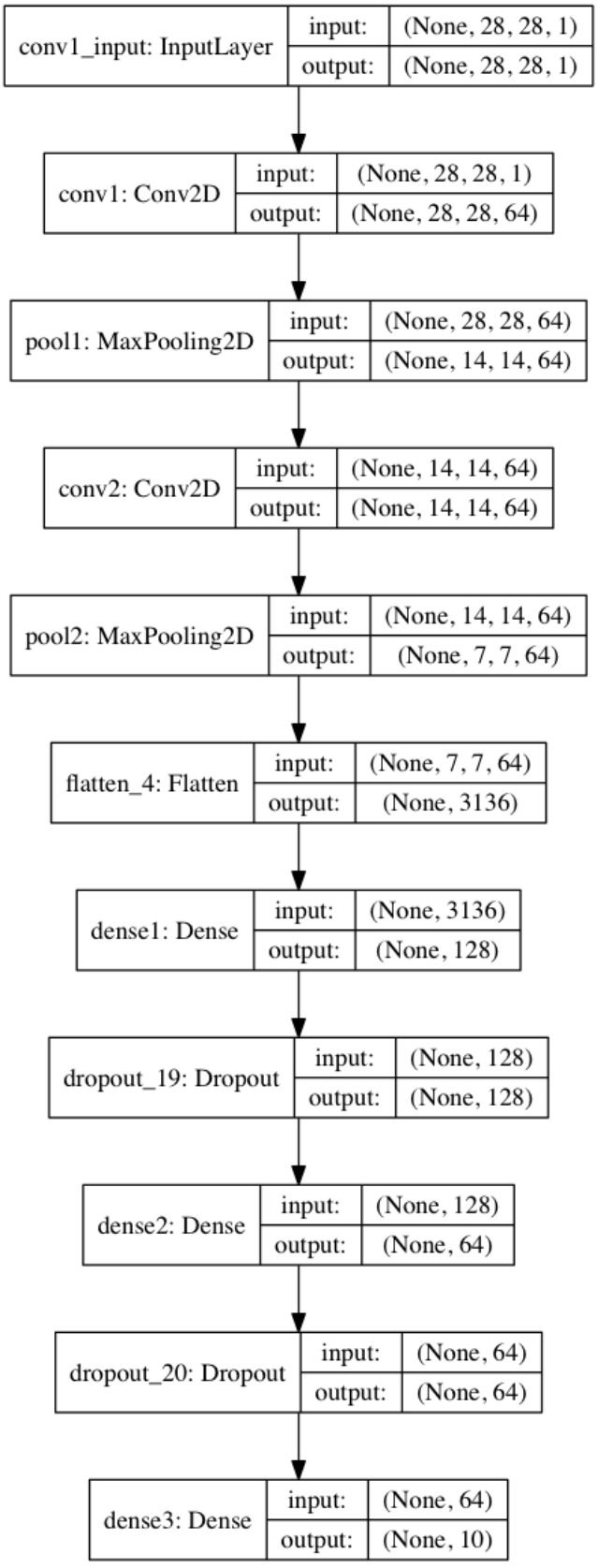

Figure B.2: Modified LeNet5 architecture

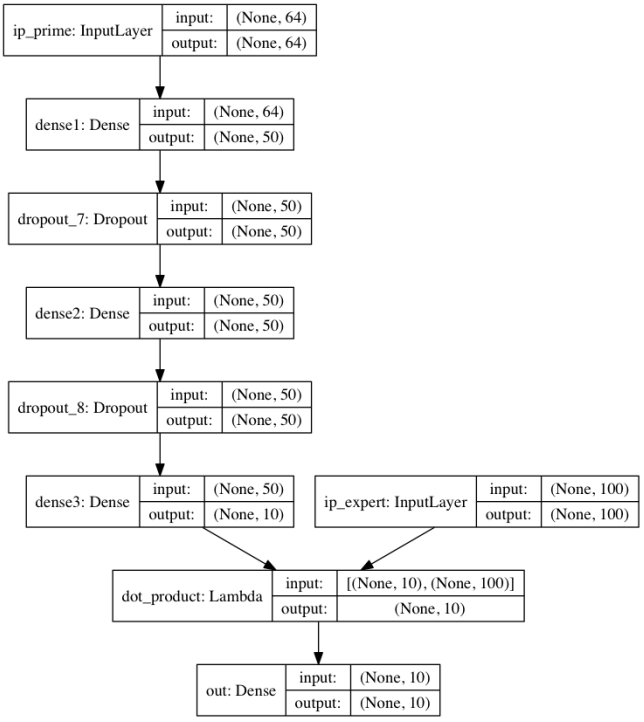

Figure B.3: Gater architecture for MNIST dataset

References

[1] Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of svms for very large scale problems. *Neural Comput.*, 14(5):1105–1114, May 2002.

[2] Shlomo E. Chazan, Jacob Goldberger, and Sharon Gannot. Speech enhancement using a deep mixture of experts. *CoRR*, abs/1703.09302, 2017.

[3] Srinivas Gutta, Jeffrey RJ Huang, P Jonathon, and Harry Wechsler. Mixture of experts for classification of gender, ethnic origin, and pose of human faces. *IEEE Transactions on neural networks*, 11(4):948–960, 2000.

[4] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[5] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23:1177–1193, 2012.

[6] M. I. Jordan. Hierarchical mixtures of experts and the em algorithm. In *IEE Colloquium on Advances in Neural Networks for Control and Systems*, pages 1/1–1/3, May 1994.

[7] Ahmed Rida, Abderrahim Labbi, and Christian Pellegrini. Experts combination through density decomposition. 2007.

[8] Volker Tresp. A bayesian committee machine. *Neural Comput.*, 12(11):2719–2741, November 2000.

[9] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *CoRR*, abs/1312.4314, 2013.

[10] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Mahdieh Abbasi and Christian Gagné. Robustness to adversarial examples through an ensemble of specialists. *CoRR*, abs/1702.06856, 2017.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[15] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,

Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[17] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[18] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.