Quantum Intelligence : One Neural Network Trains Another Using Its Prior Knowledge

Poondru Prithvinath Reddy

January 13, 2020

Quantum Intelligence : One Neural Network Trains Another Using Its Prior Knowledge

Poondru Prithvinath Reddy

ABSTRACT

Artificial Neural networks are capable of learning for themselves that will be essential for the future quantum computers mainly quantum error correction and these networks outstrip other error correction strategies. The quantum information is highly sensitive to noise from its environment and needs regular quantum error correction, and this role is performed by artificial neural networks as they gather information about the state of the quantum bits. The solution is in the form of an additional neural network i.e. one neural network uses its prior knowledge of the quantum computer that is to be controlled to train another and guide towards successful quantum correction. In this paper, we propose both conventional neural network model and a novel deep network structure i.e. " Network In Network" (NIN) to address quantum error correction. The conventional convolutional layer uses linear filters followed by nonlinear activation function to scan the input. Instead, Deep NIN uses multiple micro neural network to enhance local modelling and utilize global average pooling over feature maps in the classification layer, which is less prone to overfitting than traditional fully connected layers. We demonstrated the performances with MNIST dataset and the test results are encouraging which motivates the possibility of one neural network training another via NIN.

INTRODUCTION

Artificial neural networks are computer programs that mimic the behavior of interconnected nerve cells or neurons and are capable of learning for themselves which will be essential for the operation of future quantum computers : quantum error correction. With sufficient training this approach of using artificial neural networks will outstrip other error-correction strategies.

In quantum computers, the quantum information i.e. quantum bit or qbit is highly sensitive to noise from its environment and the quantum information needs regular repairs – that is quantum error correction.

Essentially, we need to preserve a pattern representing a certain quantum state of qbit pieces and these are the quantum error correction operations. The quantum error correction role is performed by artificial neural networks as they can even outstrip correction strategies devised by intelligent human minds. However, one artificial neural network alone is not enough as it can only gather small amounts of information about the state of the quantum bits.

The solution comes in the form of an additional neural network that acts as a teacher to the first network. With its prior knowledge of the quantum computer that is to be controlled, this teacher network is able to train the other network and thus guides toward successful quantum correction.

METHODOLOGY

Artificial neural networks have had a reputation of incorporating prior knowledge, and the Bayesian models and the state of art of those – Bayesian networks – solve this problem naturally but these models have their own known limitations.
The known strategies for incorporating prior knowledge in to a neural network model (feed forward or recurrent) are as below :

- The simplest type of prior knowledge often used is weight decay. Weight decay assumes the weights come from a normal distribution with zero mean and some fixed variance. This type of prior is added as an extra term to the loss function.
- Also there are other, less straight forward methods to incorporate prior knowledge into neural networks. One of them is Data Augmentation i.e. by training the network on data perturbed by various class – preserving transformations. The data augmentation e.g. by rotation or mirroring images, it changes the image but its class (label)

remain the same. The richer set of transformations we use, the more prior information is encoded in the model through augmented dataset. The other is Regularization Loss Terms, similar to weight decay.

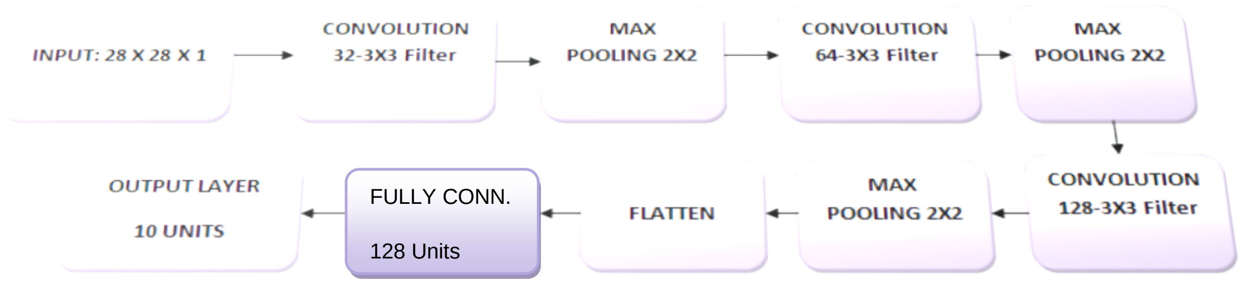
- Semantic Based Regularization (Soumali Roychoudhury et all., 2018) as a general and novel way to integrate prior knowledge into deep learning. This takes as input the prior knowledge, expressed as a collection of first-order logic clauses (FOL), where each task to be learned corresponds to a predicate in the knowledge base. Then, it translates the knowledge into a set of constraints which can be either integrated into the learning process or used in a collective classification step during the test phase. This is realized via the same backpropagation schema that runs over the expression trees of the grounded FOL clauses.

As there are different methods to incorporate prior knowledge into artificial neural networks and however, in this paper we followed the following methodology :

- Conventional Convolutional Neural Network (Sharing limited field – of – view kernels over spatial locations exploits knowledge about data which incorporates prior knowledge into the model)
- Implementation of Network In Network (NIN)

ARCHITECTURE

IMAGE CLASSIFICATION USING DEEP LEARNING



We have already heard of image or facial recognition or self—driving cars. These are real-life implementations of Convolutional Neural Networks (CNNs). We implement these deep, feed-forward artificial neural networks by overcoming overfitting with the regularization technique called "dropout".

We have used the MNIST dataset for training and testing the image processing. The **MNIST database** (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The MNIST database contains 60,000 training images and 10,000 testing images. To load the data, we first need to download the data from the link and then structure the data in a particular folder format to be able to work with it.

From the above, we can see that the training data has a shape of 60000 x 784: there are 60,000 training samples each of 784-dimensional vector. Similarly, the test data has a shape of 10000 x 784, since there are 10,000 testing samples.

The 784 dimensional vector is nothing but a 28 x 28 dimensional matrix. That's why we will be reshaping each training and testing sample from a 784 dimensional vector to a 28 x 28 x 1 dimensional matrix in order to feed the samples in to the CNN model.

As a first step, we convert each 28 x 28 image of the train and test set into a matrix of size 28 x 28 x 1 which is then fed into the network.

The Deep Artificial Neural Network

We used three convolutional layers:

- The first layer will have 32-3 x 3 filters,
- The second layer will have 64-3 x 3 filters and
- The third layer will have 128-3 x 3 filters.

In addition, there are three max-pooling layers each of size 2 x 2.

We used a RELU as our activation function which simply takes the output of max_pool and applies RELU.

Flattening layer:

The Output of a convolutional layer is a multi-dimensional Tensor. We want to convert this into a one-dimensional tensor. This is done in the Flattening layer. We simply used the reshape operation to create a single dimensional tensor.

Fully connected layer:

Now, let’s define and create a fully connected layer. Just like any other layer, we declare weights and biases as random normal distributions. In fully connected layer, we take all the inputs, do the standard z=wx+b operation on it. The Fully Connected Layer has 128 Neurons.

We added Dropout into the network to overcome the problem of overfitting to some extent and also to improve the training and validation accuracy. This way, turning off some neurons will not allow the network to memorize the training data since not all the neurons will be active at the same time and the inactive neurons will not be able to learn anything.

The Test results show good accuracy between training and validation data

IMAGE CLASSIFICATION USING DEEP LEARNING AND PRIOR KNOWLEDGE (NIN)

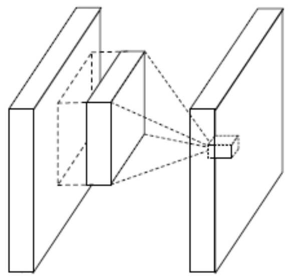
Convolutional networks, operate directly on the images as is. The filters (also called kernels) are moved across the image left to right, top to bottom as if scanning the image and weighted sum of products are calculated between the filter and subset of the image the filter is superimposing on. This is the convolution operation. What is to be noted here is that the operation is *linear*. Of course these are then passed through various other operations like non-linear activations and pooling.

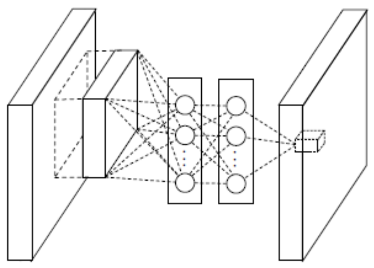
Network In Network

On the other hand, NIN has a new take on how the convolution filters are designed and how we map extracted features to class scores. This formed the basis of the inception architecture. Two new concepts were introduced in this CNN architecture design:

- **MLPconv**: Replaced linear filters with nonlinear Multi Linear Perceptrons to extract better features within the receipt field. This helped in better abstraction and accuracy.
- **Global Average Pooling**: Got rid of the fully connected layers at the end thereby reducing parameters and complexity. This was replaced by the creation of as many activation maps in the last layer as there are classes. This was followed by averaging these maps to arrive at final scores, which is passed to softmax. This is performant and more intuitive.

MLPConv


Linear Convolutional Layer


MLPConv Layer

Traditional CNN architectures use linear filters to do the convolution and extract features out of images. The early layers try to extract primitive features like lines, edges, and corners, while the later layers build on early layers and extract higher-level features like eyes, ears, nose etc. These are called latent features and there can be variations in each of those features - there can be many different variations in eyes alone.

NIN introduced the concept of having a neural network itself in place of a convolution filter. The input to this mini network would be the convolution, and the output would be the value of a neuron in the activation. Hence it does not alter the input/output characteristics of traditional filters. This mini network, called MLPconv, can then convolved over the input. The benefit of having such an arrangement is two-fold:

- It is compatible with the backpropagation logic of neural nets, thus this fits well into existing architectures of CNN's

- It can itself be a deep model leading to rich separation between latent features

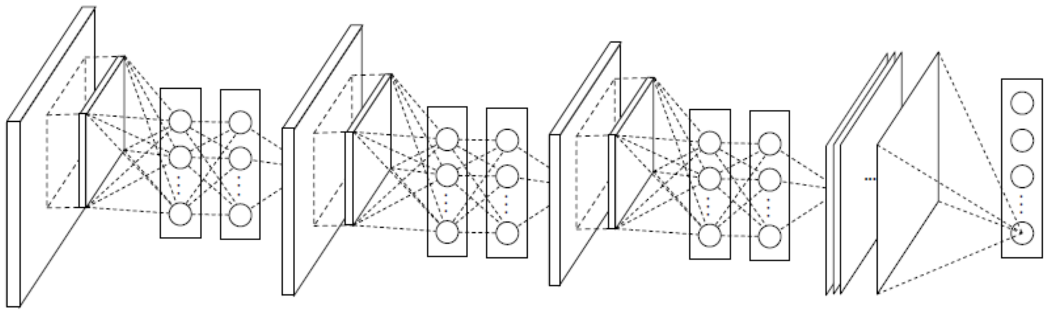
Global Average Pooling

In traditional CNN architectures, the feature maps of the last convolution layer are flattened and passed on to one or more fully connected layers, which are then passed on to softmax logistics layer for spitting out class probabilities. The issue with this approach is that it is hard to decode how the usual fully connected layers seen at the end of CNN architectures map to class probabilities. They are black boxes between the convolution layers and the classifier. They are also prone to overfitting and come with lots of parameters to train. An estimate says that the last FC layers contain 90% of the parameters of the network.

In the approach proposed by the NIN, the last MLPconv layer produces as many activation maps as the number of classes being predicted. Then, each map is averaged giving rise to the raw scores of the classes. These are then fed to a SoftMax layer to produce the probabilities, totally making FC layers redundant.

The advantages of this approach are:

- The mapping between the extracted features and the class scores is more intuitive and direct. The feature can be treated as category confidence.
- An implicit advantage is that there are no new parameters to train (unlike the FC layers), leading to less overfitting.
- Global average pooling sums out the spatial information, thus it is more robust to spatial translations of the input.



Overall Structure of Network In Network (NIN)

Thus, the above is the overall structure of NIN with global average pooling at the end.

The overall structure of NIN is a stack of mlpconv layers, on top of which lie the global average pooling layer.

The number and size of layers are as in Conventional CNN except for two mlpconv(1X1) layers between convolutional layers and avgpool(8X8) layer at the end. Also, we have applied a dropout of 0.5.

We have used the MNIST dataset for training and testing the image processing implementation in the approach as proposed by the NIN.

RESULTS

It is observed that NIN does indeed help with the gain in overall accuracy in comparison to Conventional CNN due to combination of mlpconv, globalavgpool and dropout.

CONCLUSION

Quantum Computers where the quantum states are extremely sensitive to constant interference from their environment and to combat this using active protection for quantum error correction system based on artificial neural networks, that are capable of learning by themselves, is presented. We proposed a conventional CNN and a novel deep network called “ Network In Network “ (NIN) which consists of mlpconv layers & a global average pooling layer for classification tasks. With these two components of NIN we demonstrated classification performance on MNIST dataset. The test results are encouraging and this motivates the possibility of one neural network training another via NIN.

REFERENCE

1. Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. arXiv preprint arXiv:1312.4400.