Summer Internship Project

Shubham Gupta
Priyesh Pratap Singh
Apoorva Sharma

May-July, 2019

Application of Transfer Learning on Hand Written Digits Classification

submitted to
Center for Cognitive Computing
IIIT,Allahabad
Under the guidance of
Dr. K.P. Singh

1 Introduction

1.1 Problem Statement

The problem requires us to predict the handwritten digit of any person in just one or two samples. So with the help of transfer learning, we aim to build a universal model which can correctly output any handwritten digit written by anyone.

The problem requires us to classify a handwritten digits between 0 to 9 under the limitation of very small available dataset for training. The target dataset consists of only 20 handwritten images of digits.

1.2 Motivation

Machine learning and deep learning plays an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and many more areas. Digit recognition system is the working of a machine to train itself or recognizing the digits from different sources like emails, bank cheque, papers, images, etc. and in different real-world scenarios for online handwriting recognition on computer tablets or system, recognize number plates of vehicles, processing bank cheque amounts, numeric entries in forms filled up by hand (say — tax forms) and so on.

2 Background

2.1 Neural Networks

2.1.1 What are Artificial Neural Networks?

Artificial neural network (ANN) is a computational structure inspired by a biological nervous system. An ANN consists of very simple and highly interconnected processors called neurons. The neurons are connected to each other by weighted links over which signals can pass. The process consists of data collection, analysis and processing, network structure design, number of hidden layers, number of hidden units, initializing, training the network, network simulation, weights/bias adjustments, and testing the network. Artificial neural networks are used in many different fields to process large sets of data, often providing useful analyses that allow for prediction and identification of new data. Artificial neural networks are computational structure programs consisting of interconnected processors called neurons connected by weights. They compute structural data through a process of learning and training. Data normally used by these structures have nonlinear relationships between inputs and outputs. They are used in applications such as speech recognition, imaging, control, estimation, optimization, and host of other things. They are also applied in real-world applications in the areas of finance, medical, business, mining, etc.

2.1.2 Convolutional Neural Networks

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the

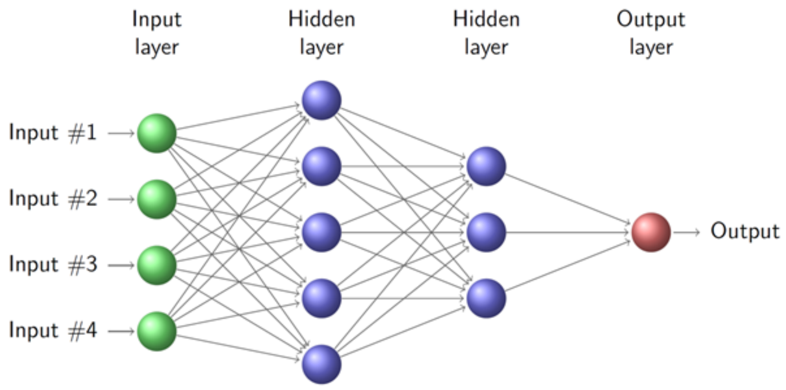


Figure 1: A simple Neural Network Representation

activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product. Mathematically, it is a sliding dot product or cross-correlation. This has significance for the indices in the matrix as it affects how weight is determined at a specific index point.

2.2 SVM

2.2.1 What is SVM?

Support Vector Machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences. A support vector machine is also known as a support vector network (SVN). More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.

Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

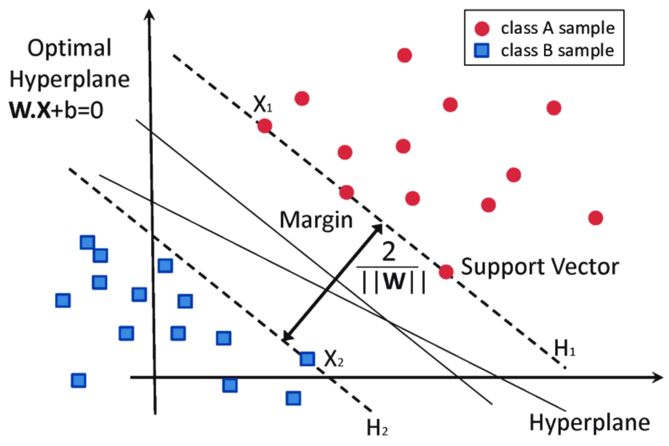


Figure 2: Figure showing how data is classified using Support Vector Machine

2.3 Transfer Learning

2.3.1 Intuitive definition

Transfer learning is a machine learning method in which a model developed for a particular task is reused as a model on another task. Transfer Learning differs from traditional Machine Learning as it is the use of pre-trained models that have been used for another task to kick start the development process on a new task or problem.

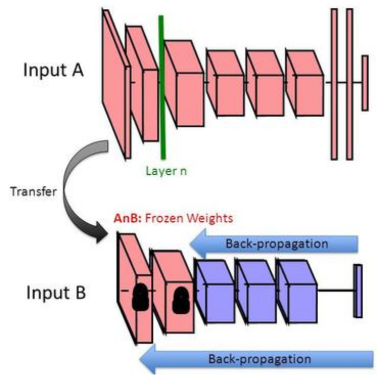


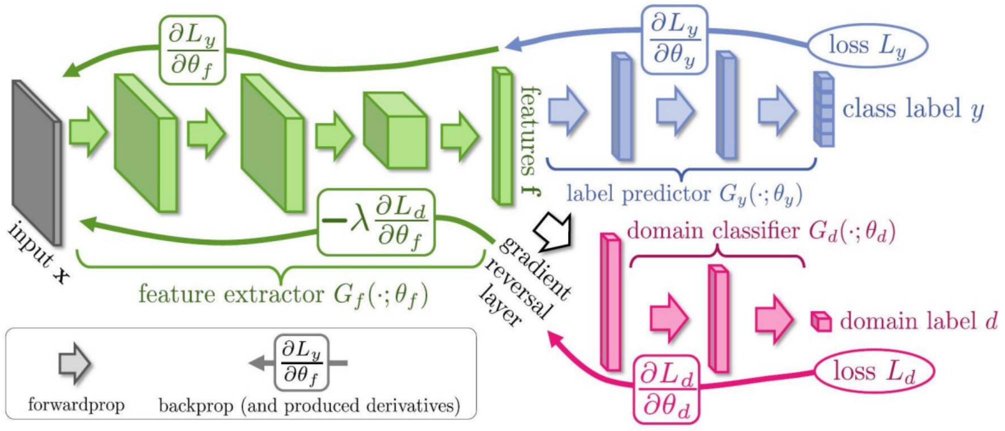
Figure 3: Figure showing how transfer learning is done in neural networks. Here weights of high level ConvNets are freezed and backpropagation is not done in these layers

2.3.2 An example

Transfer learning involves the concepts of a domain and a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space, where $X = x_1, \cdots, x_n \in \mathcal{X}$. [1]For document classification with a bag-of-words representation, $\mathcal{X}$ is the space of all document representations, x_i is the i-th term vector corresponding to some document and X is the sample of documents used for training. Given a domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ that is typically learned from the training data consisting of pairs $x_i \in X$ and $y_i \in \mathcal{Y}$. In our document classification example, $\mathcal{Y}$ is the set of all labels, i.e. True, False and y_i is either True or False. Given a source domain $\mathcal{D}_S$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, the objective of transfer learning now is to enable us to learn the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. In most cases, a limited number of labeled target examples, which is exponentially smaller than the number of labeled source examples are assumed to be available.

2.4 Domain Adaptation

Domain adaptation provides an attractive option given that labeled data of similar nature but from a different domain (e.g. synthetic images) are available. As the training progresses, the approach promotes the emergence of "deep" features that are (i) discriminative for the main learning task on the source domain and (ii) invariant with respect to the shift between the domains. This adaptation behaviour can be achieved in almost any feed-forward model by augmenting it with few standard layers and a simple new gradient reversal layer. The resulting augmented architecture can be trained using standard backpropagation.[2]



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features f
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 4: Figure showing how domain adaptation is done using a multi output model.

3 Proposed Methodology

Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layer they learn features similar to Gabor filters and color blobs. Such first-layer features appear not to specific to a particular dataset or task but are general in that they are applicable to many datasets and tasks. As finding these standard features on the first layer seems to occur regardless of the exact cost function and natural image dataset, we call these first-layer features general.

In transfer learning we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, that is, suitable to both base and target tasks, instead of being specific to the base task.

As we have to recognize handwritten digits, so we used MNIST as source dataset to extract high level features which will be very similar to high level features of our target dataset.

Since the target dataset is small, it is not a good idea to fine-tune the ConvNet due to the risk of overfitting.

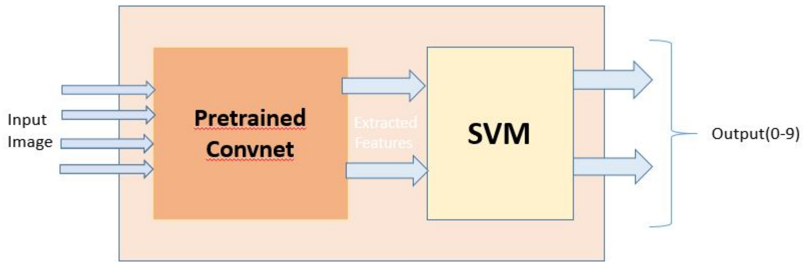


Figure 5: A basic representation of our first model

Hence, we, first trained base ConvNet using MNIST dataset then removed the fully connected layers near the end of the pretrained base ConvNet. After training of the base model (pretrained model), we froze the weights and feed it to SVM for classification.Then, We trained our whole model with small target training data using 'rbf' kernel. In this process only SVM get trained because weights of the ConvNet are frozen.

While experiementing, we built one more model(lets call it second model), by introducing the concept of domain adaptation. Here we forced our ConvNet to learn features of digits' shapes only, not color or size.For this purpose, we used a multi output model having two set of outputs, one for digit and other for domain classification. Here we have two domains, original MNIST samples and local handwritten digits. Since we want our model to not differentiate the different domains, so we tried to maximize our cost function with respect to domain classifier. And for that, we used gradient reversal technique.

The loss function for our new model is Loss Function = $\mathrm{L}_1\lambda_1 - L_2\lambda_2$ where L1 is the loss w.r.t to label classifier and L2 is w.r.t to Domain classifier.$(\lambda)1$ and $(\lambda)2$ are the respective parameters.

Here lambda1 and lambda2 are selected experimentally. For our convince we keep both equal to 1 i.e. both are equidominant.

4 Experiments

4.1 System Configuration

To efficiently train a machine learning model on massive amounts of data, data scientists or machine learning engineers often need to use specialized hardware, such as:

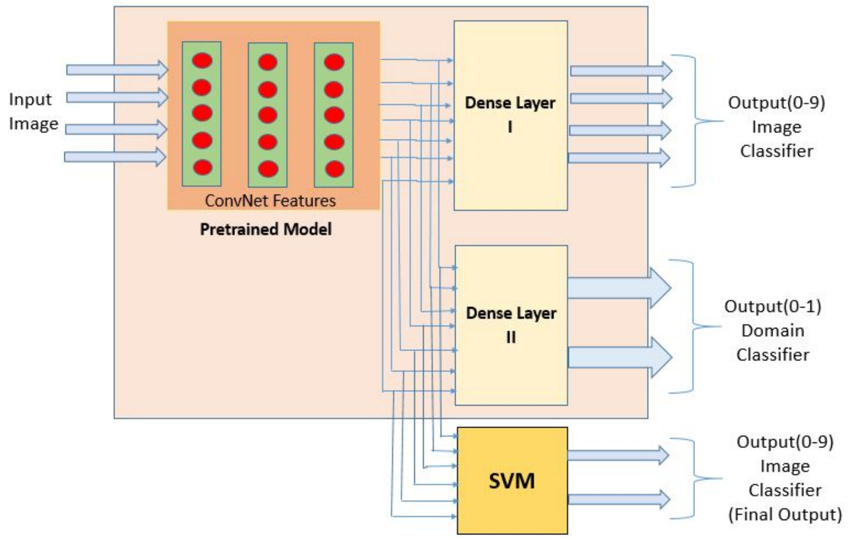


Figure 6: A basic representation of our second model

1. Graphics Processing Units - GPUs
2. Tensor Processing Units - TPUs

With these specialized hardwares, the training task that used to take weeks or months to complete now only take minutes. The problem is these hardwares are very expensive, so it requires a huge upfront cost to get started. So we used Google Colaboratory and took advantage of free GPUs and TPUs to develop our project. It has 2vCPU @ 2.2GHz with 13GB RAM.

4.2 Dataset

We have used classical mnist dataset(60,000 training and 10,000 testing samples) for building our pretrained model. We pre processed it by normalising the image vectors and reshaping it so that it can be feed to convolution network. We created our own dataset of 20 samples individually for training and testing on the new model. We recorded our digits and convert it into MNIST format by padding, cropping and normalizing the images.

4.3 Experiment Settings

We built our model in keras(github link:) with following settings:

1. First Model(Finetuning by freezing weights)
   (a) Three Conv2D layers with two MaxPooling layers in between.
   (b) Three dense layer were inserted, followed by flattening.

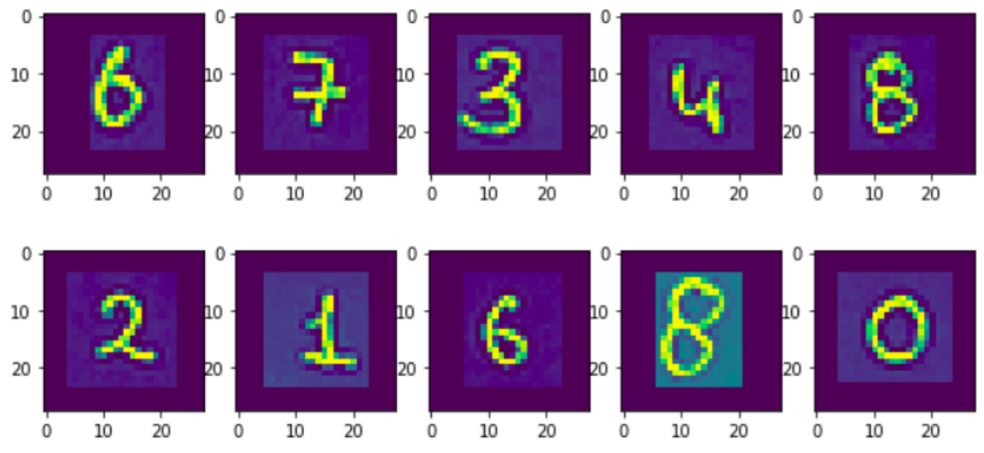


Figure 7: A sample from our self created training dataset

   (c) Activation function that we used was relu in hidden layers and softmax function for output layer.
   (d) Loss function was categorical cross entropy and optimizer was adam.

2. Second Model(Finetuning with Domain Adaptation)
   (a) Three Conv2D layers with two MaxPooling layers in between.
   (b) Three dense layer were inserted, followed by flattening.
   (c) Activation function that we used was relu in hidden layers and softmax function for output layer.
   (d) Loss function(L): L1 $(\lambda)1$-L2$(\lambda)2$, L1 and L2 are categorical cross entropy loss functions and$(\lambda)1$, $(\lambda)2$ are parameters. Adam optimizer was used for parameter updation.

5 Result and Discussion

Simple Convnet	**Train Size = 20 Test Size = 20**	**Train Size = 15 Test Size = 20**	**Train Size = 10 Test Size = 20**
Training Loss	1.6570	1.9556	1.935
Training Accuracy(%)	75	40	70
Validation Loss	2.103	2.309	2.222
Validation Accuracy(%)	35	10	25
Epochs	10	10	10

Table 1: Results obtained without using Transfer Learning

Since the accuracy that we got from our model is not very satisfactory, so we planned to introduce domain adaptation in our model.

Preained Model	
MNIST training size	60000
MNIST Validation Size	10000
Training Accuracy(%)	99.24
Training Loss	0.0231
Validation Accuracy(%)	99.09
Validation Loss	0.0320

Table 2: Results from single output model

Level 2 SVM	
Test Size = 20	Testing Accuracy(%)
Train Size = 20	70
Train Size = 15	50
Train Size = 20	65

Table 3: Final results from first model

	0-9 Label Classifier	Domain Classifier
Train Size	60020	
Test Size	10020	
Training loss	0.6043	16.1131
Testing Loss	0.0668	16.0951
Training Acc.	98.02%	0.000049
Testing Acc.	97.75	0

Table 4: Results From Multi output Model

Train Size	Test Size	Accuracy
20	20	65%

Table 5: Final Results from second model

After performing sufficient number of experiments, we observed that our domain classifier is getting biased with the MNIST domain and producing same result for different values of $(\lambda)1$ and $(\lambda)2$. We believe that this is happening because of the imbalanced dataset(60,000 vs 20).

6 Conclusion

In this paper, we present a newly configured model for Handwritten Digit Recognition using Transfer Learning with Domain Adaptation. We achieved accuracy close to 65 %. which can further be improved by using OverSampling Techniques like SMOTE[3] in domain classifier.

7 Future Scopes

As mentioned that we didn't get very good accuracy from our first model and our second model is getting biased because of unbalanced dataset. One can extend these models by performing upsampling and apply various bias-reducing techniques. One can also use well sophisticated pretrained models from autokeras or ludwig.
Humans appear to have mechanisms for deciding when to transfer information, selecting appropriate sources of knowledge, and determining the appropriate level of abstraction. It is not always clear how to make these decisions for a single machine learning algorithm, much less in general.

8 References

References

[1] Lisa Torrey, and Jude Shavlik, "Transfer Learning," in IEEE/WIC/ACM International Conference on Web Intelligence (WI); 2009.

[2] Yaroslav Ganin, and Vitor Lempitsky, "Unsupervised Domain Adaptation by Backpropogation," in 15th INTERNATIONAL CONFERENCE ON MALIGNANT LYMPHOMA.

[3] Nitesh V Chawla, Kevin W Boyer, Lawrence O Hall, W Phillip Kegelmeyer, "Synthetic Minority Over Sampling Technique," in Journal of Artificial Intelligence Research 16(2002) 321-357 .

[4] Jason Yosinski,1 Jeff Clune,2 Yoshua Bengio,3 and Hod Lipson4, "How transferable of Features in Deep Neural Networks," in NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 Pages 3320-3328 .