MATH 4432 Final Project: Transfer Learning on Fashion-MNIST and MNIST datasets

Anish Hiranandani
ahiranandani@ust.hk

Animesh Kumar Jha
akjha@ust.hk

Abstract

This project explores the phenomenon of transfer learning on 2 datasets, namely MNIST and Fashion-MNIST. We give a brief overview of the origin of Fashion-MNIST and describe the process of transfer learning. Next, our report discusses the VGG16 architecture which has been pre-trained on ImageNet, a large image recognition database. After extracting features using VGG16, we use unsupervised learning methods such as PCA and KMeans clustering to visualize these features and lastly evaluate models such as Logistic Regression, SVM, ANN, Random Forests on both the datasets.

1 Introduction

The MNIST dataset is a subset of the dataset prepared by researchers at National Institute of Standards and Technology which contains 60,000 training images and 10,000 testing images. Each grayscale image is a 28x28 pixels representation of one of the 10 handwritten digits in our number system. In the original paper, an accuracy of 97% was obtained using a Support Vector Machine(SVM). This is a very high accuracy rate, without any doubt. Further, with the advent of newer and more complex machine learning algorithms, one of the more recent papers reports an accuracy of 99.77% using the hierarchical system of convolutional neural networks(CNN).

According to researchers at Zalando, MNIST is consistently one of the first datasets that people interested in machine learning work on. However, over time this dataset has been overused and might not represent the true performance of any new research conducted. Several claims have been made about the fact that the MNIST dataset is too easy and most pairs of handwritten digits can be told apart by seeing just 1 pixel. Machine Learning enthusiasts often over-train models to this dataset. To solve this issue and motivate people to move away to a better and harder dataset, Zalando created a new dataset called Fashion MNIST which contains exactly the same number of training and testing images, in the same

28x28 pixel representation with the same number of classes (10). The classes are as follows:
{t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker bag, ankle boot}. [1]

The intention of researchers at Zalando is very clear: to replace MNIST with Fashion MNIST. It shares the same structure and size as the original MNIST and is intended to be harder.

2 Objective

The objective of this project is to use the method of transfer learning to extract features from the Fashion MNIST dataset, carry out dimensionality reduction via Principal Component Analysis(PCA) and clustering to visualize these features and lastly apply simple machine learning techniques such as Random Forest, K-Nearest Neighbors, Logistic Regression, Linear Discriminant Analysis and Quadratic Discriminant Analysis to these features to train and evaluate our dataset.

Some of the scientific questions proposed for this project are as follows:

1. How good are the accuracy scores of the machine learning algorithms after extracting features using transfer learning?
2. Is it really true that the MNIST dataset is overused and "too easy" as claimed by people all around the world? We take a subset of the original MNIST training set to evaluate this.
3. If true, can we find some possible reasons/explanation for this during this project?

To facilitate the answers of the above questions, we will carry out the exact same procedures on the MNIST dataset and compare the results.

3 Overview on Transfer Learning

Transfer Learning refers to the phenomenon of using knowledge gained while solving one problem and using it to solve another related problem in the future. In this project, we have chosen to use VGG16. VGG16 is a very deep convolutional network built for large scale image recognition. VGG was trained on ImageNet (an image database of over 14 million images having over 20 thousand ambiguous categories). [2]

VGG16 consists of 16 convolutional layers and is often used as a baseline feature extractor. The architecture looks like this:

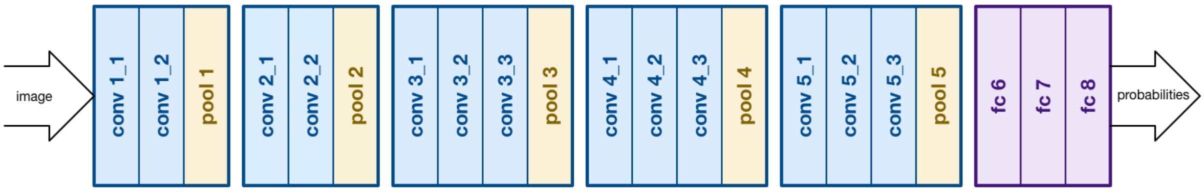

Figure 1: VGG16 Architecture

The VGG16 architecture looks like this (after excluding the topmost layer):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 48, 48, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 48, 48, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 48, 48, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 24, 24, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 24, 24, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 24, 24, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 6, 6, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Overview of VGG16 Architecture with parameters

To extract features, we exclude the topmost layer and calculate the 2^{nd} last layer in the network using the weights pre-trained on ImageNet. For each image in our dataset (Fashion-MNIST/MNIST) this would give us a tensor of size (1, 1, 512) which can be flattened and viewed as 512 features. This network has 14,714,688 trainable parameters.

4 Data Preprocessing

The Fashion-MNIST dataset was downloaded from Kaggle. The training set has 60,000 images and the testing set has 10,000 images. The MNIST dataset was downloaded from Kaggle.

The following were the preprocessing steps for this project:

1. Since VGG16 accepts images to have 3 channels(RGB), we converted the 1 channel grayscale images to RGB format using the dstack function in the numpy library.
2. Since VGG16 accepts images in a 48x48 pixel format, we resized the original 28x28 images into this format.
3. The VGG16 weights pre-trained on ImageNet was not downloadable from github and had to be manually downloaded in h5 format and saved on the machine. Further, the VGG script had to be downloaded and modified to support accessing the weights from the local machine rather than downloading from github.

5 Feature Extraction and Visualization

The training and testing data were passed through the VGG16 network and the resulting features were flattened to create a vector of size 512 for each image. Using unsupervised learning methods such as PCA and K-Means clustering, we reduced dimensionality to 3 and plotted the scatter plots for both datasets.

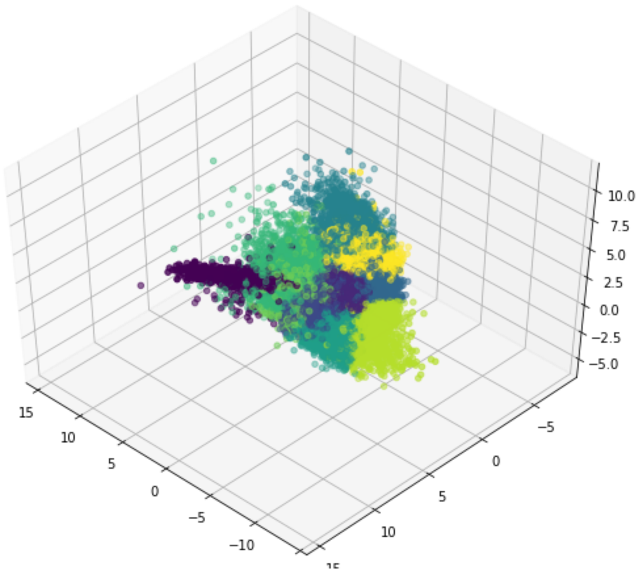

Figure 3: Fashion-MNIST scatter plot after KMeans clustering (Training)

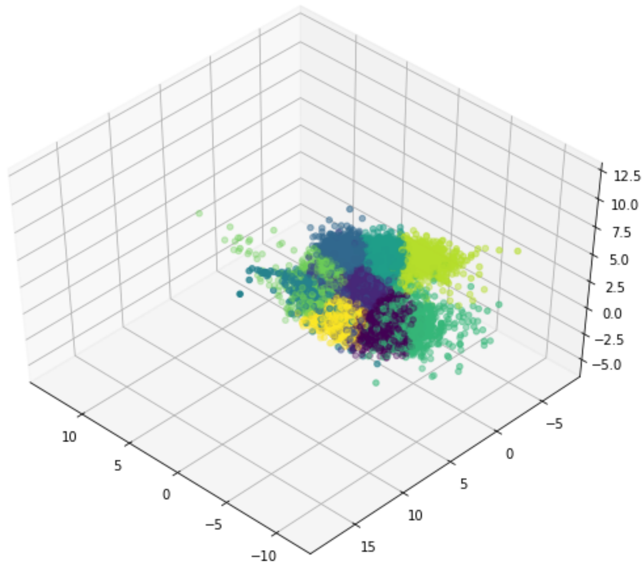

Figure 4: Fashion-MNIST scatter plot after KMeans clustering (Testing)

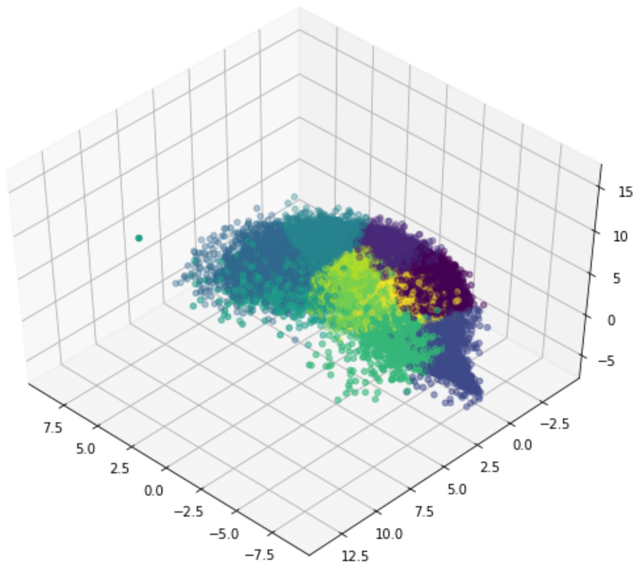

Figure 4: MNIST scatter plot after KMeans clustering (Training)

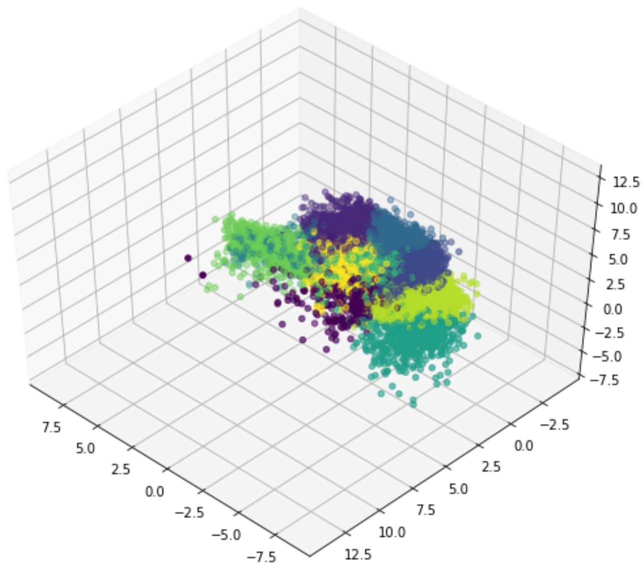

Figure 6: MNIST scatter plot after KMeans clustering (Testing)

Note that z-score normalization was done to map the feature data to a zero mean and unit variance distribution as a preprocessing step before applying PCA.

The difference in the scatter plots for the 2 datasets is quite striking.

The original MNIST dataset has a more well-formed (almost semi-circular in 3D) distribution for the training set. One can easily identify at least 8 classes distinctly separated with ease. The testing set on the other hand, is slightly more random, but still vaguely represents a semi-circle in 3D. Perhaps, the difference in number of data points contributes to this observation.

The Fashion-MNIST dataset is definitely more scattered than the MNIST dataset. We can segregate up-to 7 classes with ease but yet, we notice some overlaps. The testing set shows similar characteristics and does not clearly separate the classes. This elegant procedure of dimensionality reduction and clustering in a classification problem gives us some insights on why MNIST is regarded as an "easy dataset" and researchers feel a strong need to replace it.

6 Classification using extracted features

The next step is to use the features extracted using the pre-trained VGG network and evaluate results on both datasets.
We chose 7 algorithms (with different hyper parameters) for this purpose. The table below gives a summary of the results.

Algorithm used	**Testing Accuracy on Fashion-MNIST**	**Testing Accuracy on MNIST**
Logistic Regression	0.7302	0.8031
Random Forest (25 trees, gini)	0.7641	0.9117
Random Forest (50 trees, gini)	0.7691	0.9172
Random Forest (100 trees, gini)	0.7749	0.9213
Random Forest (25 trees, entropy)	0.7657	0.9115
Random Forest (50 trees, entropy)	0.7713	0.9165
Random Forest (100 trees, entropy)	0.7772	0.9219
KNearestNeighbour (3 neighbours)	0.7580	0.8800

KNearestNeighbour (5 neighbours)	0.7640	0.8768
Neural Network (64, 32) hidden layers	0.7548	0.8211
Perceptron	0.5997	0.6882
SVM (Linear)	0.7026	0.7992
SVM (Polynomial degree 3)	0.7401	0.8121
SVM (Linear with 15 PCA components)	0.7770	0.8443

We obtain a high level of accuracy on the MNIST dataset as expected, even on simpler models after transfer learning. The performance on Fashion-MNIST is impressive, given that this dataset is meant to be challenging and we have not trained a deep convolutional neural network from scratch. The fact that the MNIST training dataset was smaller (~48,000 samples), is also a testament to the fact that the Fashion-MNIST is a more challenging dataset. We see that Random Forest classifier performed the best for both Fashion-MNIST and MNIST. This was unexpected as we predicted that a neural network would perform better. Linear SVM with all 512 features took a very long time to train and hence, we tried reducing the number of features to 15 using PCA and then applying SVM. There was a 7% and 4% increase in accuracy respectively both datasets and the time taken was reduced significantly. These results indicate that Fashion-MNIST would indeed be an interesting dataset to experiment algorithms on and see if we can achieve similar levels of accuracy as MNIST.

7 Contributions

Anish has contributed by doing transfer learning using VGG16 on the Fashion-MNIST and MNIST datasets, dimensionality reduction using PCA and KMeans clustering for visualization and writing the report. Animesh has contributed by running the machine learning models mentioned in the table to both the datasets.

8 References

[1] Research.zalando.com. (2018). Fashion MNIST.
https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/

[2] Medium. (2018). CNNs Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more
https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5