Classification of MNIST Images using VGG16 embeddings and SVM

Team Members

1.	Nikhil Heda	01FB14ECS130
2.	Nisha Daga	01FB14ECS135
3.	Pooja Kabber	01FB14ECS147

Problem Definition

Image classification is the process of identifying or predicting the class or category of an image after extracting the features necessary for the classification. These features can be extracted manually, using feature extraction algorithms or, automatically, using Deep Learning techniques.

Dataset Used

The aim of this project is to classify images from the MNIST (Modified National Institute of Standards and Technology) dataset. The MNIST database is a large database of handwritten digits that contains 60,000 training images and 10,000 testing images. These images were taken from NIST's databases which consists of images of digits written by high school students and employees of the United States Census Bureau.

Techniques Used

Leveraging transfer learning to generate embeddings for images and feeding the features extracted to a SVM model. In order to classify the images from the MNIST database, we use a pre-trained Deep Learning model called VGG16. We use the VGG16 model to extract features and then use the image embeddings that the model produces with the classification algorithm, Support Vector Machines. This process of using a pre-trained model is called Transfer Learning.

Similar to the Word2Vec neural network model that produces word embeddings for text, the entire VGG16 model is used here to generate image embeddings that can then be used with a faster classification algorithm like Support Vector Machines.

Design

The design of this project is as follows -
Firstly, the images from the database are resized and converted to a color space that VGG16
requires and are given to the model through the input layer. The embeddings that are gotten from the

output of the model are stored in .npy files so that they do not have to be calculated every time they are needed.

Fig. 1 shows the architecture of VGG16.

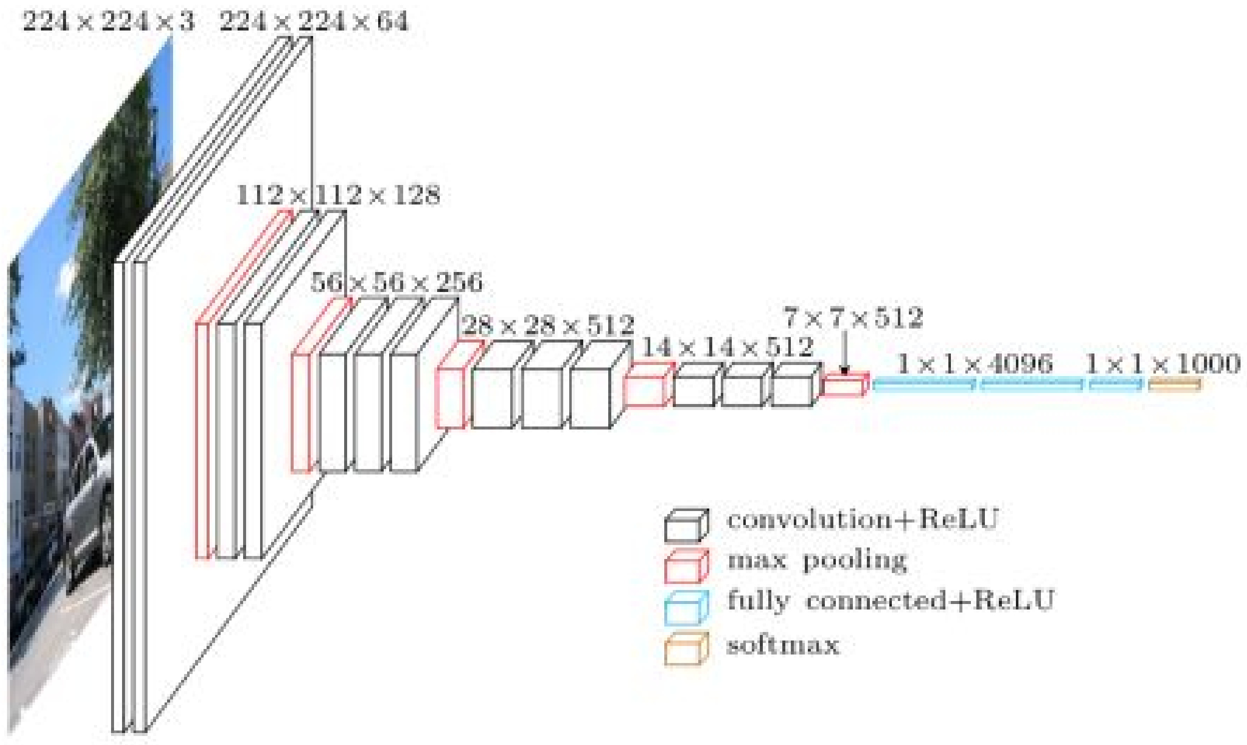

Fig. 1 Architecture of VGG16

The embeddings from the pre-trained model are then input to three SVM models of the following summaries:

- Kernel: ‘RBF’ C = 1

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

- Kernel: ‘Linear’C = 1000

```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

- Kernel: ‘Linear’Gamma = 0.0001 C = 1000

```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

- Kernel: ‘RBF’ C = 1000

```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

- Kernel: Linear C = 1

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Algorithm

- Load dataset into memory
- Converting into RGB since VGG16 expects each pixel to have three dimensions
- Extract VGG16 features to get the model
- Train the model on training dataset
- Store the trained model to be used later
- Creating a SVM model using the output of VGG16
- Calculating the accuracy by using a simple formula -

Accuracy = (Images classified correctly / Total # of Images) * 100

Test Results

Model	Accuracy (in %)	Observations
SVM with rbf kernel and C = 1	52	● Image embedding are in high dimensional space hence are likely to be linearly separable, and rbf kernel may overfit the data leading to less accuracy. ● Higher C also implies overfitting on training data.
SVM with rbf kernel and C = 1000	56.9	
SVM with linear kernel and C = 1	92	● Linear Kernel outperforms Radial Basis Kernel, as data is high dimensional and more likely to be linearly separable, also we use a low gamma value (large variance) which gives us slightly better performance. ● Lower C is probably underfitting the data slightly as it may misclassify some data points due to larger margin hyperplane. ● Higher C value, evaluates to a slightly better decision boundary and higher accuracy.
SVM with linear kernel and C = 1000	93	
SVM with linear kernel, gamma = 0.0001 and C = 1000	93	