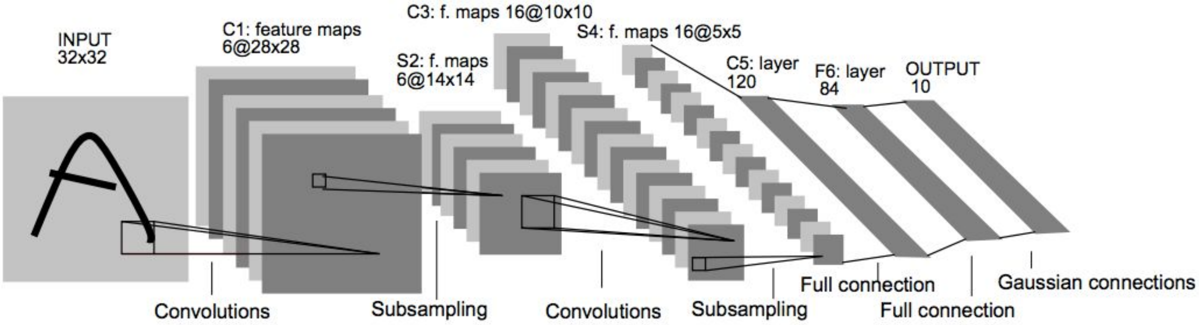
Comparative analysis of traditional neural network and LeNet on MNIST dataset

Name: Shrinidhi Anil Varna
Roll No: 171CO145
Course: Computer Vision (CO468)

Architecture 1: LeNet-5



LeNet-5 is our latest convolutional network designed for handwritten and machine-printed character recognition.

The dataset used for training the LeNet-5 model is available here.

Code Snippets:

1. **Load data from MNIST dataset**

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", reshape=False)
X_train, y_train           = mnist.train.images, mnist.train.labels
X_validation, y_validation = mnist.validation.images, mnist.validation.labels
X_test, y_test             = mnist.test.images, mnist.test.labels
```

2. **Pad 28x28 images by 2 on all sides to make it 32x32 (input shape required for LeNet-5)**

```
import numpy as np

# Pad images with 0s
X_train      = np.pad(X_train, ((0,0),(2,2),(2,2),(0,0)), 'constant')
X_validation = np.pad(X_validation, ((0,0),(2,2),(2,2),(0,0)), 'constant')
X_test       = np.pad(X_test, ((0,0),(2,2),(2,2),(0,0)), 'constant')
```

3. **Shuffle data**

```
from sklearn.utils import shuffle

X_train, y_train = shuffle(X_train, y_train)
```

4. **LeNet-5 architecture model**
The model is compiled with sparse cross-entropy as the loss function and fitted with the training and validation features (X) and labels (y). There is a callback function set to check for early stopping and saving checkpoints. Validation loss is monitored by the callback function. During the training process (model.fit()), a timer is set which measures the time taken for fitting the model with the features and the labels.

```
EPOCHS = 30
BATCH_SIZE = 100
rate = 0.001

optimizer = tf.train.AdamOptimizer(learning_rate = rate)
lenet_5_model = tf.keras.models.Sequential([
   tf.keras.layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh', input_shape =
X_train[0].shape, padding='same'), # Conv2D layer 1
   tf.keras.layers.AveragePooling2D(), # Sub sampling layer 1
```

```
    tf.keras.layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh', padding='valid'), # Conv2D layer 2
    tf.keras.layers.AveragePooling2D(), # Sub sampling layer 2
    tf.keras.layers.Flatten(), # Flatten
    tf.keras.layers.Dense(120, activation='tanh'), # Fully connected layer
    tf.keras.layers.Dense(84, activation='tanh'), # Fully connected layer
    tf.keras.layers.Dense(10, activation='softmax') # Output layer
])
```

5. **Time taken to train = 237.5430166721344 seconds**

6. **Testing the model**

   metrics_dict = lenet_5_model.evaluate(X_test, y_test)

   Test accuracy = 98.4399 %
   Test loss = 0.04776516

7. **Confusion matrix**

```
[[ 970,    0,    0,    1,    0,    1,    4,    0,    3,    1],
 [ 0, 1122,    1,    2,    0,    0,    2,    4,    4,    0],
 [ 3,    0, 1021,    0,    1,    0,    0,    7,    0,    0],
 [ 0,    0,    1, 1004,    0,    0,    0,    2,    3,    0],
 [ 0,    1,    2,    2,  959,    0,    3,    2,    0,   13],
 [ 2,    0,    0,   15,    0,  870,    3,    1,    0,    1],
 [ 2,    2,    1,    1,    6,    3,  942,    0,    1,    0],
 [ 1,    0,    6,    3,    0,    0,    0, 1014,    1,    3],
 [ 2,    0,    5,    5,    0,    1,    1,    2,  955,    3],
 [ 2,    2,    1,    4,    4,    3,    0,    6,    0,  987]]
```

8. **Precision and Recall**

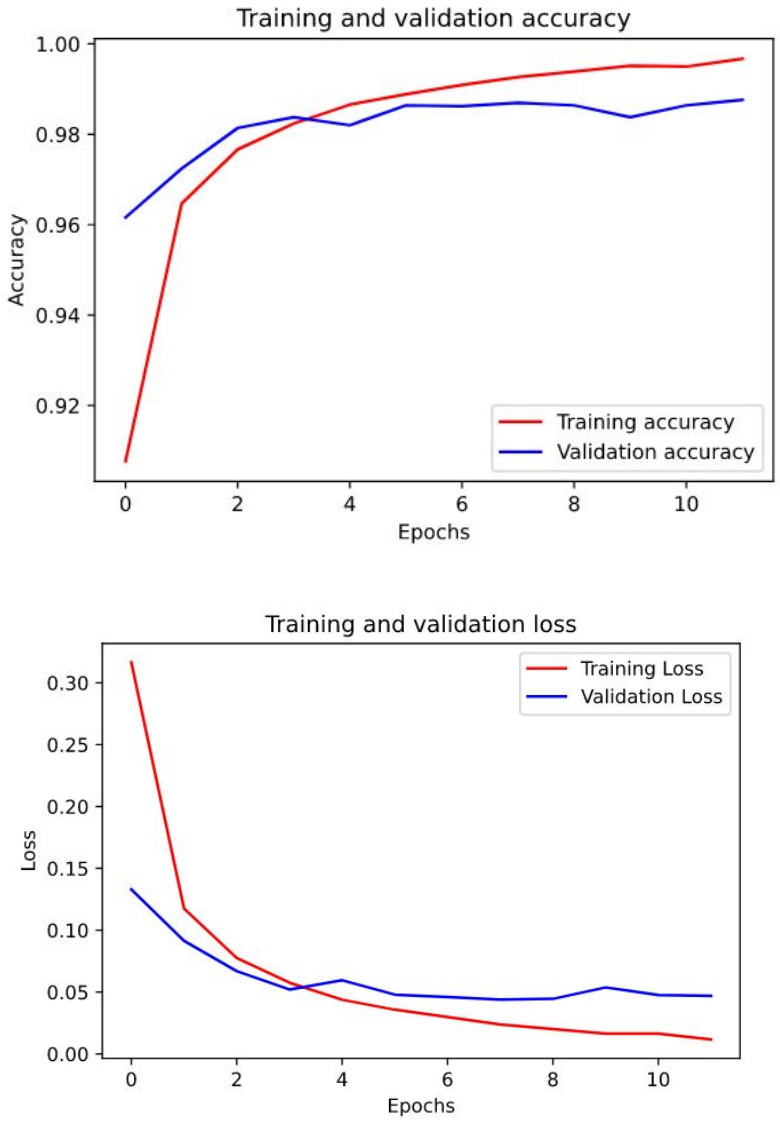
   from sklearn.metrics import precision_score, recall_score

   precision_recall = metrics.classification_report(y_test, np.argmax(y_pred, axis = 1))

	Precision	Recall
0	0.99	0.99
1	1.00	0.99
2	0.98	0.99

3	0.97	0.99
4	0.99	0.98
5	0.99	0.98
6	0.99	0.98
7	0.98	0.99
8	0.99	0.98
9	0.98	0.98
Weighted average	0.98	0.98

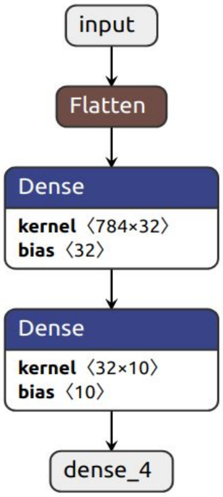
PTO



Graphs plotted for the training process of LeNet-5

Architecture 2: Traditional Neural Network

PTO



1. **Load data**

```
mnist = input_data.read_data_sets("MNIST_data/", reshape=False)
X_train, y_train           = mnist.train.images, mnist.train.labels
X_validation, y_validation = mnist.validation.images, mnist.validation.labels
X_test, y_test             = mnist.test.images, mnist.test.labels
```

2. **Traditional Neural Network Architecture**
The model is compiled with sparse cross-entropy as the loss function and fitted with the training and validation features (X) and labels (y). There is a callback function set to check for early stopping and saving checkpoints. Validation loss is monitored by the callback function. During the training process (model.fit()), a timer is set which measures the time taken for fitting the model with the features and the labels.

```
ANN_model = tf.keras.models.Sequential ([
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(32, activation='tanh'), # Fully connected layer,
   tf.keras.layers.Dense(10, activation='softmax', input_shape = X_train[0].shape)])
```

Time taken to train = 22.29641580581665 seconds

3. **Testing the model**

```
metrics_dict = ANN_model.evaluate(X_test, y_test)
```

Test accuracy = 96.42 %
Test loss = 0.1244659

4. **Confusion matrix**

```
[[ 963,    1,    2,    0,    1,    2,    6,    2,    3,    0],
 [ 0, 1116,    5,    1,    0,    1,    3,    2,    7,    0],
 [ 4,    0,  993,   10,    2,    1,    5,    5,   12,    0],
 [ 2,    0,    6,  980,    1,    7,    0,    7,    6,    1],
 [ 0,    0,    6,    0,  947,    1,    7,    2,    5,   14],
 [ 3,    1,    3,   15,    2,  843,    8,    1,   14,    2],
 [ 5,    3,    5,    1,    4,    6,  930,    1,    3,    0],
 [ 0,    3,   15,    8,    3,    0,    1,  987,    4,    7],
 [ 4,    1,    4,    9,    4,    4,    2,    3,  941,    2],
 [ 4,    4,    0,   11,   24,    5,    4,   10,    5,  942]]
```

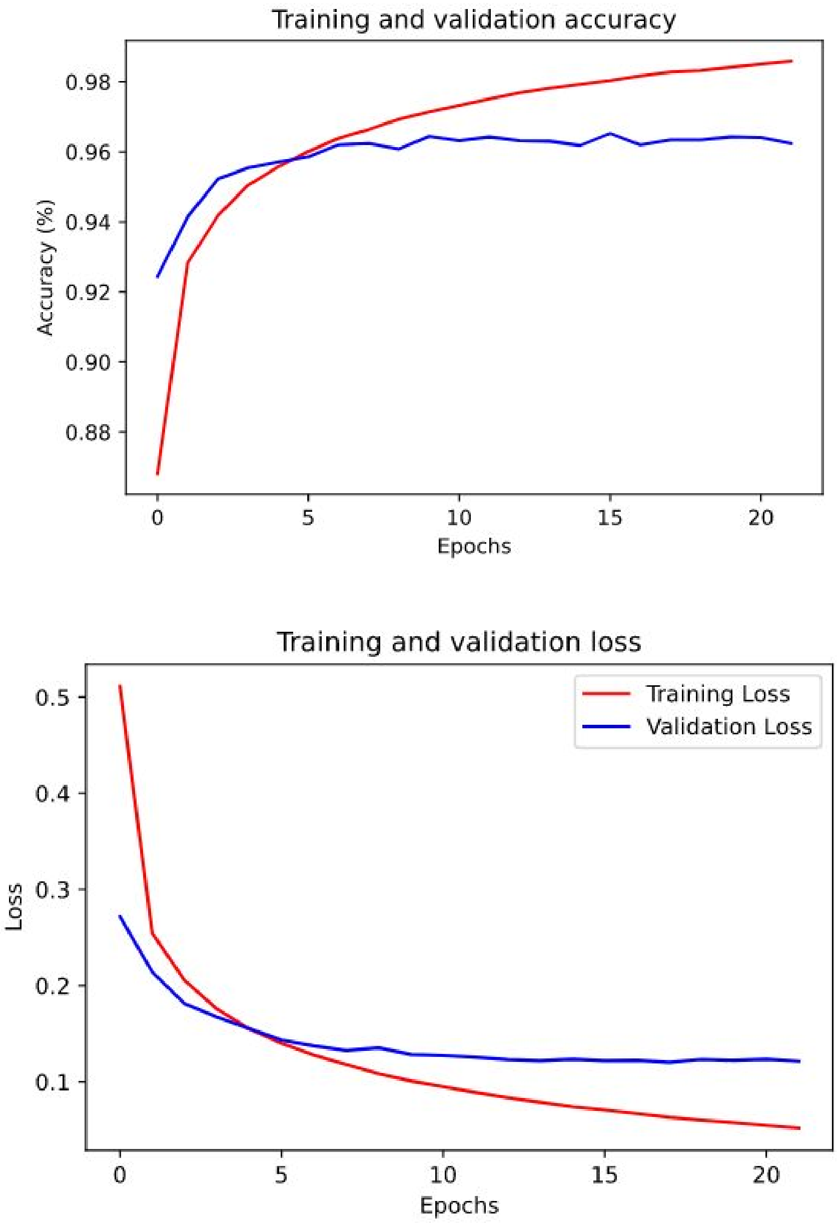
5. Precision and recall

```
from sklearn.metrics import precision_score, recall_score
```

```
precision_recall = metrics.classification_report(y_test, np.argmax(y_pred, axis = 1))
```

	Precision	Recall
0	0.98	0.98
1	0.99	0.98
2	0.96	0.96
3	0.95	0.97
4	0.96	0.96
5	0.97	0.95
6	0.96	0.97
7	0.97	0.96
8	0.94	0.97
9	0.97	0.93
Weighted average	0.96	0.96

PTO



Graphs plotted for the training process of LeNet-5

Comparison of all the metrics

Metric Name	LeNet-5	Traditional Neural Network
Test accuracy (%)	98.4399	96.42

Test loss	0.04776516	0.1244659
Time taken to train (sec)	237.5430166721344	22.29641580581665
Precision	0.98	0.96
Recall	0.98	0.96