Lab Report 4

Convolution Neural Networks

Goutham Krishnan

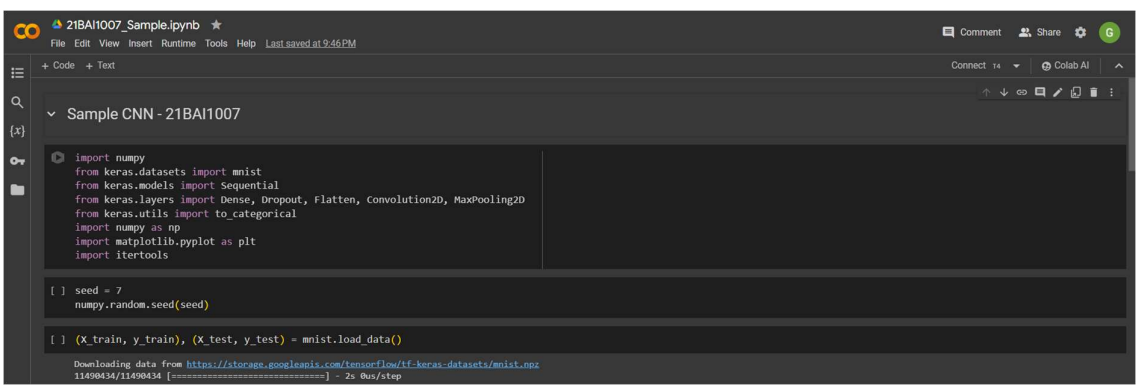
21BAI1007

Aim

1. To create a simple CNN model for MNIST Digit classification dataset.
2. To compare 5 different CNN models for the MNIST Dataset.
3. Visualize filter and feature for the CNN model.
4. Create a CNN for another customized dataset (CIFAR-10).
5. Compare CNN with a Fully Connected Dense model.

Observations

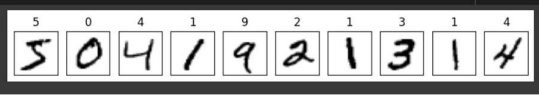
For aim 1: To create a simple CNN model for MNIST Digit classification dataset.



21BAI1007_Sample.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:46 PM

```python
fig, ax = plt.subplots(1, 10, figsize=(10, 10))
for i in range(0, 10):
    ax[i].xaxis.set_visible(False)
    ax[i].yaxis.set_visible(False)
    ax[i].set_title(y_train1[i])
    ax[i].imshow(x_train[i], cmap=plt.cm.binary)

def baseline_model():
    model = Sequential()
    model.add(Convolution2D(32, (5, 5), padding='valid', strides=(1,1), input_shape=(28,28,1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(1152, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```



21BAI1007_Sample.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:46 PM

```python
model = baseline_model()
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbose=2)
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```
Epoch 1/10
300/300 - 8s - loss: 0.1750 - accuracy: 0.9475 - val_loss: 0.0575 - val_accuracy: 0.9796 - 8s/epoch - 26ms/step
Epoch 2/10
300/300 - 2s - loss: 0.0488 - accuracy: 0.9848 - val_loss: 0.0442 - val_accuracy: 0.9861 - 2s/epoch - 8ms/step
Epoch 3/10
300/300 - 3s - loss: 0.0322 - accuracy: 0.9897 - val_loss: 0.0388 - val_accuracy: 0.9878 - 3s/epoch - 10ms/step
Epoch 4/10
300/300 - 2s - loss: 0.0215 - accuracy: 0.9929 - val_loss: 0.0403 - val_accuracy: 0.9871 - 2s/epoch - 8ms/step
Epoch 5/10
300/300 - 2s - loss: 0.0183 - accuracy: 0.9940 - val_loss: 0.0367 - val_accuracy: 0.9884 - 2s/epoch - 8ms/step
Epoch 6/10
300/300 - 3s - loss: 0.0127 - accuracy: 0.9959 - val_loss: 0.0330 - val_accuracy: 0.9897 - 3s/epoch - 9ms/step
Epoch 7/10
300/300 - 3s - loss: 0.0118 - accuracy: 0.9962 - val_loss: 0.0338 - val_accuracy: 0.9906 - 3s/epoch - 11ms/step
Epoch 8/10
300/300 - 2s - loss: 0.0095 - accuracy: 0.9969 - val_loss: 0.0434 - val_accuracy: 0.9886 - 2s/epoch - 8ms/step
Epoch 9/10
300/300 - 2s - loss: 0.0098 - accuracy: 0.9967 - val_loss: 0.0397 - val_accuracy: 0.9894 - 2s/epoch - 6ms/step
Epoch 10/10
300/300 - 2s - loss: 0.0072 - accuracy: 0.9974 - val_loss: 0.0439 - val_accuracy: 0.9889 - 2s/epoch - 6ms/step
CNN Error: 1.11%
```

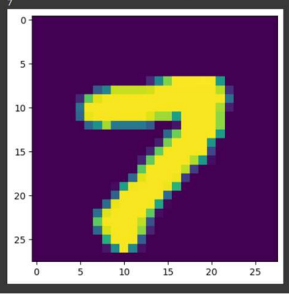
21BAI1007_Sample.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:46 PM

Predicting an image - 21BAI1007

```python
num = 79;
plt.imshow(x_test[num])
testSample = x_test[num].reshape(1, x_test[num].shape[0], 28, 1).astype('float32')
pred = model.predict(testSample)
for i in range(0, 10):
  if pred[0][i]==1:
    print(i);
```

```
1/1 [==============================] - 0s 106ms/step
7
```



For aim 2: To compare 5 different CNN models for the MNIST Dataset.

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comparing 5 different Models on the MNIST Dataset - 21BAI1007

```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Convolution2D, MaxPooling2D
from keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
import itertools
```

```
seed = 7
numpy.random.seed(seed)
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
```

```
X_train = X_train/255
X_test = X_test/255
```

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]
(x_train, y_train1), (x_test, y_test1) = mnist.load_data()
```

```
X_train[1].shape
```

```
(28, 28, 1)
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Model 1 - 21BAI1007

```
fig, ax = plt.subplots(1, 10, figsize=(10, 10))
for i in range(0, 10):
    ax[i].xaxis.set_visible(False)
    ax[i].yaxis.set_visible(False)
    ax[i].set_title(y_train1[i])
    ax[i].imshow(x_train[i], cmap=plt.cm.binary)

def baseline_model():
    model = Sequential()
    model.add(Convolution2D(32, (5, 5), padding='valid', strides=(1,1), input_shape=(28,28,1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(1152, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
model = baseline_model()
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbose=2)
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```
Epoch 1/10
300/300 - 7s - loss: 0.1704 - accuracy: 0.9481 - val_loss: 0.0519 - val_accuracy: 0.9833 - 7s/epoch - 25ms/step
Epoch 2/10
300/300 - 2s - loss: 0.0508 - accuracy: 0.9842 - val_loss: 0.0445 - val_accuracy: 0.9854 - 2s/epoch - 6ms/step
Epoch 3/10
300/300 - 2s - loss: 0.0327 - accuracy: 0.9894 - val_loss: 0.0318 - val_accuracy: 0.9894 - 2s/epoch - 6ms/step
Epoch 4/10
300/300 - 2s - loss: 0.0228 - accuracy: 0.9929 - val_loss: 0.0439 - val_accuracy: 0.9857 - 2s/epoch - 6ms/step
Epoch 5/10
300/300 - 2s - loss: 0.0190 - accuracy: 0.9938 - val_loss: 0.0319 - val_accuracy: 0.9878 - 2s/epoch - 6ms/step
Epoch 6/10
300/300 - 2s - loss: 0.0123 - accuracy: 0.9961 - val_loss: 0.0392 - val_accuracy: 0.9870 - 2s/epoch - 7ms/step
Epoch 7/10
300/300 - 2s - loss: 0.0122 - accuracy: 0.9958 - val_loss: 0.0337 - val_accuracy: 0.9897 - 2s/epoch - 6ms/step
Epoch 8/10
300/300 - 2s - loss: 0.0089 - accuracy: 0.9969 - val_loss: 0.0335 - val_accuracy: 0.9895 - 2s/epoch - 7ms/step
Epoch 9/10
300/300 - 2s - loss: 0.0092 - accuracy: 0.9969 - val_loss: 0.0445 - val_accuracy: 0.9879 - 2s/epoch - 7ms/step
Epoch 10/10
300/300 - 2s - loss: 0.0094 - accuracy: 0.9968 - val_loss: 0.0287 - val_accuracy: 0.9916 - 2s/epoch - 6ms/step
CNN Error: 0.84%
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Second Model - 21BAI1007

```python
def model2():
    model = Sequential()
    model.add(Convolution2D(64, (5, 5), padding='valid', strides=(1,2), input_shape=(28,28,1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model2 = model2()
model2.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbose=2)
scores = model2.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/10
300/300 - 5s - loss: 0.2776 - accuracy: 0.9120 - val_loss: 0.0809 - val_accuracy: 0.9759 - 5s/epoch - 18ms/step
Epoch 2/10
300/300 - 2s - loss: 0.0716 - accuracy: 0.9788 - val_loss: 0.0477 - val_accuracy: 0.9853 - 2s/epoch - 5ms/step
Epoch 3/10
300/300 - 2s - loss: 0.0508 - accuracy: 0.9852 - val_loss: 0.0413 - val_accuracy: 0.9864 - 2s/epoch - 5ms/step
Epoch 4/10
300/300 - 2s - loss: 0.0366 - accuracy: 0.9895 - val_loss: 0.0410 - val_accuracy: 0.9878 - 2s/epoch - 5ms/step
Epoch 5/10
300/300 - 2s - loss: 0.0339 - accuracy: 0.9898 - val_loss: 0.0413 - val_accuracy: 0.9880 - 2s/epoch - 5ms/step
Epoch 6/10
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Third model - 21BAI1007

```python
def model3():
    model = Sequential()
    model.add(Convolution2D(32, (3, 3), padding='valid', strides=(1,1), input_shape=(28, 28, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    #model.add(Dropout(0.2))
    #model.add(Dense(32, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model3 = model3()
model3.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size=64, verbose=1)
scores = model3.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/20
938/938 [==============================] - 8s 6ms/step - loss: 0.1879 - accuracy: 0.9421 - val_loss: 0.0496 - val_accuracy: 0.9832
Epoch 2/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0500 - accuracy: 0.9844 - val_loss: 0.0538 - val_accuracy: 0.9845
Epoch 3/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0365 - accuracy: 0.9884 - val_loss: 0.0365 - val_accuracy: 0.9884
Epoch 4/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0292 - accuracy: 0.9905 - val_loss: 0.0282 - val_accuracy: 0.9919
Epoch 5/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0229 - accuracy: 0.9928 - val_loss: 0.0354 - val_accuracy: 0.9883
Epoch 6/20
938/938 [==============================] - 5s 6ms/step - loss: 0.0185 - accuracy: 0.9942 - val_loss: 0.0302 - val_accuracy: 0.9912
Epoch 7/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0152 - accuracy: 0.9952 - val_loss: 0.0293 - val_accuracy: 0.9912
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Fourth Model - ALEXNET - 21BAI1007

```python
def model4():
    model = Sequential()
    model.add(Convolution2D(96, (3,3), strides=1, input_shape=(28,28,1), activation='relu'))
    model.add(MaxPooling2D((3,3), strides=2))
    model.add(Convolution2D(256, (5,5), padding='same', activation='relu'))
    model.add(MaxPooling2D((3,3), strides=2))
    model.add(Convolution2D(384, (3,3), padding='same', activation='relu'))
    model.add(Convolution2D(384, (3,3), padding='same', activation='relu'))
    model.add(Convolution2D(256, (3,3), padding='same', activation='relu'))
    model.add(MaxPooling2D((3,3), strides=2))
    model.add(Flatten())
    model.add(Dense(4096, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(4096, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model4 = model4()
model4.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=30, batch_size=64, verbose=1)
scores = model4.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 3/30
938/938 [==============================] - 22s 23ms/step - loss: 0.0480 - accuracy: 0.9876 - val_loss: 0.0424 - val_accuracy: 0.9889
Epoch 4/30
938/938 [==============================] - 22s 23ms/step - loss: 0.0403 - accuracy: 0.9895 - val_loss: 0.0411 - val_accuracy: 0.9893
Epoch 5/30
938/938 [==============================] - 22s 24ms/step - loss: 0.0351 - accuracy: 0.9914 - val_loss: 0.0590 - val_accuracy: 0.9866
Epoch 6/30
938/938 [==============================] - 22s 24ms/step - loss: 0.0301 - accuracy: 0.9924 - val_loss: 0.0663 - val_accuracy: 0.9843
Epoch 7/30
938/938 [==============================] - 22s 23ms/step - loss: 0.0287 - accuracy: 0.9925 - val_loss: 0.0333 - val_accuracy: 0.9907
Epoch 8/30
938/938 [==============================] - 22s 23ms/step - loss: 0.0233 - accuracy: 0.9942 - val_loss: 0.0256 - val_accuracy: 0.9939
Epoch 9/30
938/938 [==============================] - 22s 23ms/step - loss: 0.0251 - accuracy: 0.9940 - val_loss: 0.0309 - val_accuracy: 0.9928
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect T4 Colab AI

Fifth Model - VGG16 - 21BAI1007

```
[ ] from keras.layers import ZeroPadding2D

    def model5():
        model = Sequential()
        model.add(Input((28, 28, 1)))
        model.add(ZeroPadding2D((2,2)))
        model.add(Convolution2D(64, (3,3), strides=1, padding='same', activation='relu'))
        model.add(Convolution2D(64, (3,3), strides=1, padding='same', activation='relu'))
        model.add(MaxPooling2D((2,2), strides=2))
        model.add(Convolution2D(128, (3,3), strides=1, padding='same', activation='relu'))
        model.add(Convolution2D(128, (3,3), strides=1, padding='same', activation='relu'))
        model.add(MaxPooling2D((2,2), strides=2))
        model.add(Convolution2D(256, (3,3), strides=1, padding='same', activation='relu'))
        model.add(Convolution2D(256, (3,3), strides=1, padding='same', activation='relu'))
        model.add(Convolution2D(256, (3,3), strides=1, padding='same', activation='relu'))
        model.add(MaxPooling2D((2,2), strides=2))
        model.add(Flatten())
        model.add(Dense(4096, activation='relu'))
        model.add(Dense(128, activation='relu'))
        model.add(Dense(10, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model

    model5 = model5()
    model5.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=64, verbose=1)
    scores = model5.evaluate(X_test, y_test, verbose=0)
    print("CNN Error: %.2f%%" % (100-scores[1]*100))
    print("Accuracy: %.2f%%" % (scores[1]*100))
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect T4 Colab AI

```
[ ]
    Epoch 1/15
    938/938 [==============================] - 24s 22ms/step - loss: 0.1972 - accuracy: 0.9367 - val_loss: 0.0410 - val_accuracy: 0.9872
    Epoch 2/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0550 - accuracy: 0.9843 - val_loss: 0.0381 - val_accuracy: 0.9891
    Epoch 3/15
    938/938 [==============================] - 21s 22ms/step - loss: 0.0406 - accuracy: 0.9882 - val_loss: 0.0446 - val_accuracy: 0.9864
    Epoch 4/15
    938/938 [==============================] - 21s 22ms/step - loss: 0.0343 - accuracy: 0.9899 - val_loss: 0.0362 - val_accuracy: 0.9894
    Epoch 5/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0308 - accuracy: 0.9910 - val_loss: 0.0398 - val_accuracy: 0.9896
    Epoch 6/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0260 - accuracy: 0.9928 - val_loss: 0.0342 - val_accuracy: 0.9903
    Epoch 7/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0226 - accuracy: 0.9936 - val_loss: 0.0272 - val_accuracy: 0.9916
    Epoch 8/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0206 - accuracy: 0.9943 - val_loss: 0.0332 - val_accuracy: 0.9897
    Epoch 9/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0203 - accuracy: 0.9945 - val_loss: 0.0319 - val_accuracy: 0.9926
    Epoch 10/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0177 - accuracy: 0.9950 - val_loss: 0.0294 - val_accuracy: 0.9933
    Epoch 11/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0173 - accuracy: 0.9952 - val_loss: 0.0357 - val_accuracy: 0.9900
    Epoch 12/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0176 - accuracy: 0.9954 - val_loss: 0.0317 - val_accuracy: 0.9917
    Epoch 13/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0137 - accuracy: 0.9962 - val_loss: 0.0409 - val_accuracy: 0.9907
    Epoch 14/15
    938/938 [==============================] - 20s 22ms/step - loss: 0.0145 - accuracy: 0.9959 - val_loss: 0.0369 - val_accuracy: 0.9920
    Epoch 15/15
    938/938 [==============================] - 20s 21ms/step - loss: 0.0133 - accuracy: 0.9963 - val_loss: 0.0501 - val_accuracy: 0.9914
    CNN Error: 0.86%
    Accuracy: 99.14%
```

21BAI1007_Q1_Compare.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect T4 Colab AI

Results - 21BAI1007

First Model:

Accuracy: 99.16%

Error %: 0.84%

Second Model

Accuracy: 98.79%

Error %: 1.21%

Third Model

Accuracy: 99.27%

Error %: 0.73%

Fourth Model

Accuracy: 99.17%

Error %: 0.83%

Fifth Model

Accuracy: 99.14%

Error %: 0.86%

Model	Accuracy	Error
First Model	99.16%	0.84%
Second Model	98.79%	1.21%
Third Model	99.27%	10.73%
Fourth Model	99.17%	0.63%
Fifth Model	99.14%	0.86%

For aim 3: Visualize filter and feature for the CNN model.

21BAI1007-visualize.ipynb

File Edit View Insert Runtime Tools Help Last saved at February 28

+ Code + Text

Visualizing filters and features - 21BAI1007

```
[ ] import numpy
    from keras.datasets import mnist
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, Flatten, Convolution2D, MaxPooling2D
    from keras.utils import to_categorical
    import numpy as np
    import matplotlib.pyplot as plt
    import itertools
```

```
[ ] seed = 7
    numpy.random.seed(seed)
```

```
[ ] (X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
[ ] X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
    X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
```

```
[ ] X_train = X_train/255
    X_test = X_test/255
```

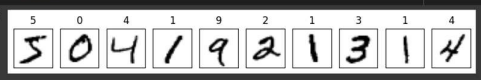
```
[ ] y_train = to_categorical(y_train)
    y_test = to_categorical(y_test)
    num_classes = y_test.shape[1]
    (x_train, y_train1), (x_test, y_test1) = mnist.load_data()
```

21BAI1007-visualize.ipynb

File Edit View Insert Runtime Tools Help Last saved at February 28

+ Code + Text

```
fig, ax = plt.subplots(1, 10, figsize=(10, 10))
for i in range(0, 10):
    ax[i].xaxis.set_visible(False)
    ax[i].yaxis.set_visible(False)
    ax[i].set_title(y_train1[i])
    ax[i].imshow(x_train[i], cmap=plt.cm.binary)

def baseline_model():
    model = Sequential()
    model.add(Convolution2D(32, (5, 5), padding='valid', strides=(1,1), input_shape=(28,28,1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(1152, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```




```
[ ] model = baseline_model()
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbose=2)
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

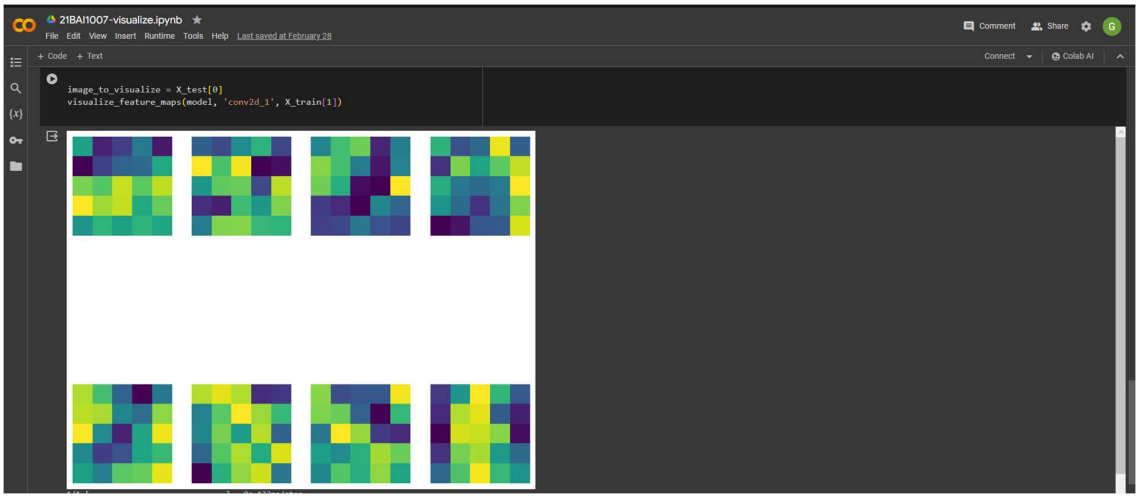
```
Epoch 1/10
300/300 - 105s - loss: 0.1722 - accuracy: 0.9491 - val_loss: 0.0498 - val_accuracy: 0.9840 - 105s/epoch - 350ms/step
Epoch 2/10
300/300 - 96s - loss: 0.0491 - accuracy: 0.9844 - val_loss: 0.0439 - val_accuracy: 0.9859 - 96s/epoch - 321ms/step
Epoch 3/10
300/300 - 95s - loss: 0.0323 - accuracy: 0.9897 - val_loss: 0.0288 - val_accuracy: 0.9908 - 95s/epoch - 317ms/step
Epoch 4/10
300/300 - 95s - loss: 0.0244 - accuracy: 0.9920 - val_loss: 0.0365 - val_accuracy: 0.9887 - 95s/epoch - 317ms/step
Epoch 5/10
300/300 - 96s - loss: 0.0179 - accuracy: 0.9939 - val_loss: 0.0316 - val_accuracy: 0.9907 - 96s/epoch - 320ms/step
Epoch 6/10
300/300 - 94s - loss: 0.0144 - accuracy: 0.9955 - val_loss: 0.0377 - val_accuracy: 0.9879 - 94s/epoch - 313ms/step
```

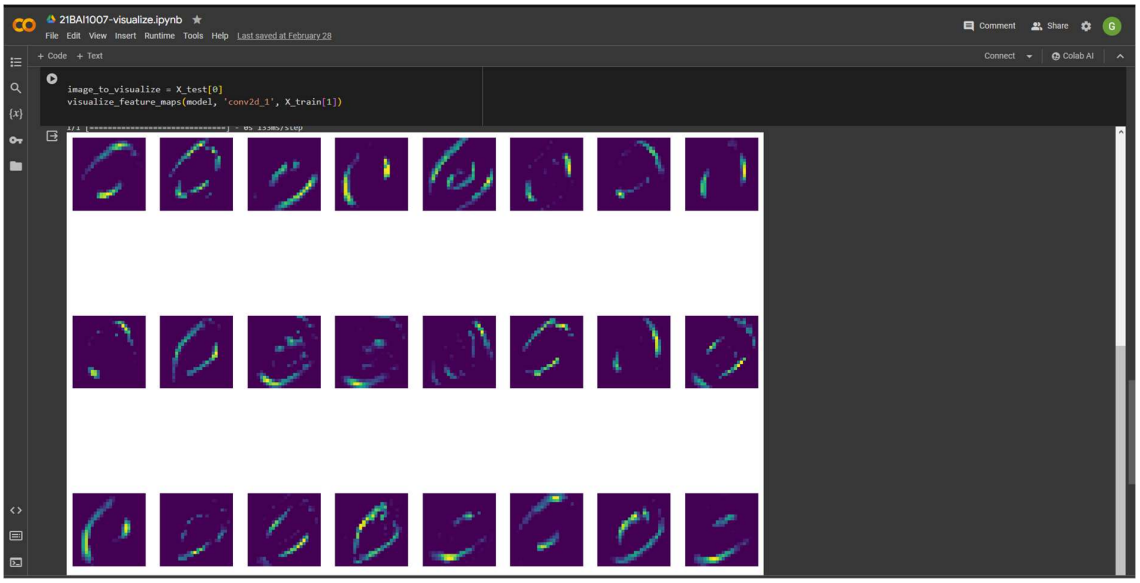
21BAI1007-visualize.ipynb

File Edit View Insert Runtime Tools Help Last saved at February 28

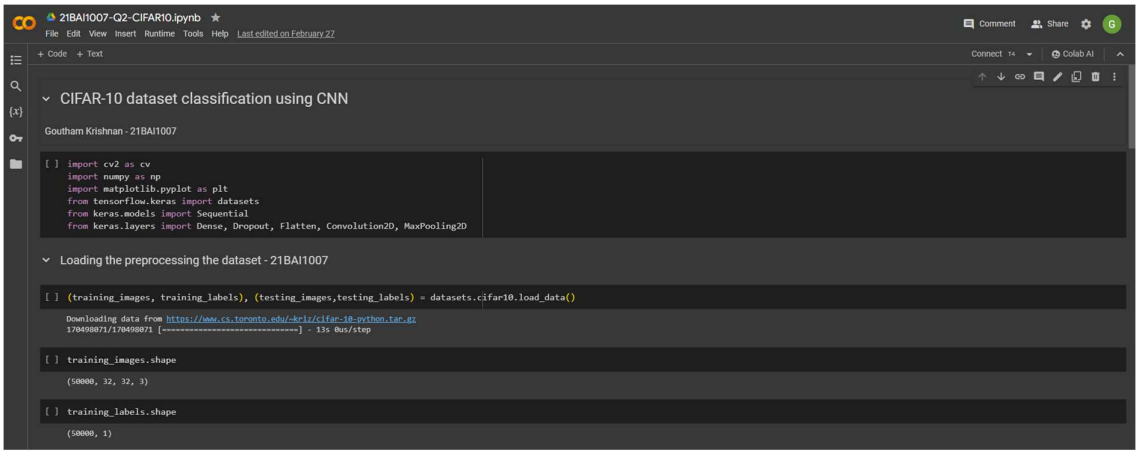
+ Code + Text

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model

def visualize_filters(model, layer_name, n_filters=8, filter_size=(3, 3), figsize=(10, 10)):
    layer = model.get_layer(layer_name)
    filters = layer.get_weights()[0]

    if len(filters.shape) == 3:
        filters = np.expand_dims(filters, axis=-1)

    plt.figure(figsize=figsize)

    for i in range(n_filters):
        plt.subplot(n_filters // 4, 4, i + 1)
        plt.imshow(filters[:, :, 0, i], cmap='viridis')
        plt.axis('off')

    plt.show()

def visualize_feature_maps(model, layer_name, input_image):
    layer = model.get_layer(layer_name)
    intermediate_model = Model(inputs=model.input, outputs=layer.output)

    input_image = np.expand_dims(input_image, axis=0)
    feature_maps = intermediate_model.predict(input_image)

    n_features = feature_maps.shape[-1]
    plt.figure(figsize=(15, 15))

    for i in range(n_features):
        plt.subplot(n_features // 8, 8, i + 1)
        plt.imshow(feature_maps[0, :, :, i], cmap='viridis')
        plt.axis('off')

    plt.show()


visualize_filters(model, 'conv2d_1')
```





For aim 4: Create a CNN for another customized dataset (CIFAR-10).

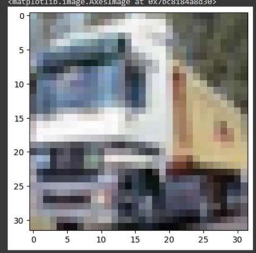


21BAI1007-Q2-CIFAR10.ipynb

File Edit View Insert Runtime Tools Help Last edited on February 27

```
plt.imshow(training_images[1])
```

```
<matplotlib.image.AxesImage at 0x7bc8184a8d30>
```



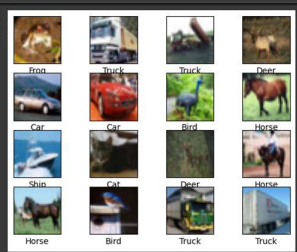
```
training_images, testing_images = training_images / 255, testing_images / 255
```

```
class_names = ['Plane', 'Car', 'Bird', 'Cat','Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']
```

```
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(training_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[training_labels[i][0]])

plt.show()
```

21BAI1007-Q2-CIFAR10.ipynb

File Edit View Insert Runtime Tools Help Last edited on February 27




```
X_train = training_images
y_train = training_labels
X_test = testing_images
y_test = testing_labels
```

```
X_train.shape
```

```
(50000, 32, 32, 3)
```

```
X_test.shape
```

```
(10000, 32, 32, 3)
```

21BAI1007-Q2-CIFAR10.ipynb

File Edit View Insert Runtime Tools Help Last edited on February 27

Creating the model - 21BAI1007

```
def model():
    model = Sequential()
    model.add(Convolution2D(32, (3, 3), padding='valid', strides=(1,1), input_shape=(32, 32, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    #model.add(Dropout(0.2))
    #model.add(Dense(32, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = model()
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size=64, verbose=1)
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/20
782/782 [==============================] - 7s 7ms/step - loss: 1.6148 - accuracy: 0.4096 - val_loss: 1.3142 - val_accuracy: 0.5311
Epoch 2/20
782/782 [==============================] - 4s 5ms/step - loss: 1.2400 - accuracy: 0.5575 - val_loss: 1.1809 - val_accuracy: 0.5787
Epoch 3/20
782/782 [==============================] - 4s 5ms/step - loss: 1.0741 - accuracy: 0.6225 - val_loss: 1.0384 - val_accuracy: 0.6336
Epoch 4/20
782/782 [==============================] - 5s 6ms/step - loss: 0.9690 - accuracy: 0.6612 - val_loss: 1.0095 - val_accuracy: 0.6502
Epoch 5/20
782/782 [==============================] - 4s 5ms/step - loss: 0.8900 - accuracy: 0.6919 - val_loss: 0.9654 - val_accuracy: 0.6681
Epoch 6/20
782/782 [==============================] - 4s 5ms/step - loss: 0.8327 - accuracy: 0.7082 - val_loss: 0.8900 - val_accuracy: 0.6923
Epoch 7/20
782/782 [==============================] - 5s 6ms/step - loss: 0.7813 - accuracy: 0.7277 - val_loss: 0.8892 - val_accuracy: 0.6919
Epoch 8/20
782/782 [==============================] - 4s 6ms/step - loss: 0.7349 - accuracy: 0.7444 - val_loss: 0.8537 - val_accuracy: 0.7016
Epoch 9/20
782/782 [==============================] - 4s 5ms/step - loss: 0.6976 - accuracy: 0.7565 - val_loss: 0.8533 - val_accuracy: 0.7106
Epoch 10/20
```

21BAI1007-Q2-CIFAR10.ipynb

```
782/782 [==============================] - 5s 7ms/step - loss: 0.4424 - accuracy: 0.8426 - val_loss: 0.9724 - val_accuracy: 0.7156
Epoch 19/20
782/782 [==============================] - 5s 6ms/step - loss: 0.4182 - accuracy: 0.8523 - val_loss: 0.9788 - val_accuracy: 0.7091
Epoch 20/20
782/782 [==============================] - 4s 6ms/step - loss: 0.3980 - accuracy: 0.8590 - val_loss: 0.9911 - val_accuracy: 0.7179
CNN Error: 28.21%
Accuracy: 71.79%
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(loss, accuracy*100,"%")
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.9911 - accuracy: 0.7179
0.991096556186676 71.78999781608582 %
```

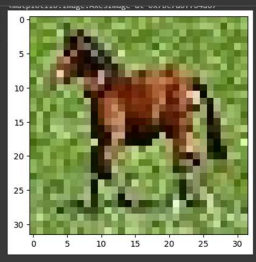
Predictions - 21BAI1007

The image of a horse is given as input and the prediction is retrieved

```
### Testing with a random image of a horse
img = cv.imread('resized_horse.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
```

```
plt.imshow(img, cmap=plt.cm.binary)
```

21BAI1007-Q2-CIFAR10.ipynb



```
prediction = model.predict(np.array([img])/255)
```

```
1/1 [==============================] - 0s 424ms/step
```

```
prediction
```

```
array([[4.6116588e-10, 9.2871959e-09, 1.3066985e-07, 5.9190910e-07,
        3.8527597e-03, 7.8939847e-05, 7.4427282e-03, 9.8862457e-01,
        1.4382377e-07, 1.0928353e-07]], dtype=float32)
```

21BAI1007-Q2-CIFAR10.ipynb

```
prediction
```

```
array([[4.6116588e-10, 9.2871959e-09, 1.3066985e-07, 5.9190910e-07,
        3.8527597e-03, 7.8939847e-05, 7.4427282e-03, 9.8862457e-01,
        1.4382377e-07, 1.0928353e-07]], dtype=float32)
```

```
index = np.argmax(prediction)
```

```
index
```

```
7
```

```
print(f'Prediction : {class_names[index]}')
```

```
Prediction : Horse
```

For aim 5: Compare CNN with a fully connected dense model.

21BAI1007_CNNvsDense.ipynb

Comparing CNN and Dense Network on CIFAR-10 dataset

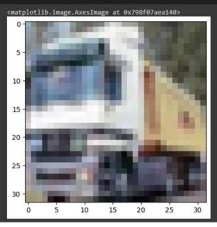
Goutham Krishnan 21BAI1007

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from keras import datasets
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Convolution2D, MaxPooling2D
```

```
(training_images, training_labels), (testing_images,testing_labels) = datasets.cifar10.load_data()
```

```
plt.imshow(training_images[1])
```

```
<matplotlib.image.AxesImage at 0x798f07aea140>
```



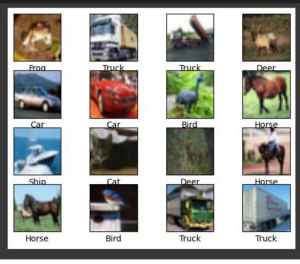
21BAI1007_CNNvsDense.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:36 AM

```
[ ] training_images, testing_images = training_images / 255, testing_images / 255
```

```
[ ] class_names = ['Plane', 'Car', 'Bird', 'Cat','Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']
```

```
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(training_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[training_labels[i][0]])

plt.show()
```



21BAI1007_CNNvsDense.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:36 AM

```
[ ] X_train = training_images
    y_train = training_labels
    X_test = testing_images
    y_test = testing_labels
```

Creating the model using CNN - 21BAI1007

```
[ ] def model():
        model = Sequential()
        model.add(Convolution2D(32, (3, 3), padding='valid', strides=(1,1), input_shape=(32, 32, 3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(Flatten())
        model.add(Dense(64, activation='relu'))
        #model.add(Dropout(0.2))
        #model.add(Dense(32, activation='relu'))
        model.add(Dense(10, activation='softmax'))
        model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model

    model = model()
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, verbose=1)
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("CNN Error: %.2f%%" % (100-scores[1]*100))
    print("Accuracy: %.2f%%" % (scores[1]*100))
```

21BAI1007_CNNvsDense.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:36 AM

```
    Epoch 3/10
[ ] 782/782 [==============================] - 60s 77ms/step - loss: 1.0756 - accuracy: 0.6210 - val_loss: 1.0429 - val_accuracy: 0.6308
    Epoch 4/10
    782/782 [==============================] - 60s 77ms/step - loss: 0.9750 - accuracy: 0.6575 - val_loss: 1.1180 - val_accuracy: 0.6185
    Epoch 5/10
    782/782 [==============================] - 61s 78ms/step - loss: 0.9025 - accuracy: 0.6837 - val_loss: 0.9618 - val_accuracy: 0.6635
    Epoch 6/10
    782/782 [==============================] - 58s 75ms/step - loss: 0.8429 - accuracy: 0.7028 - val_loss: 0.8888 - val_accuracy: 0.6945
    Epoch 7/10
    782/782 [==============================] - 62s 79ms/step - loss: 0.7982 - accuracy: 0.7221 - val_loss: 0.8778 - val_accuracy: 0.6956
    Epoch 8/10
    782/782 [==============================] - 59s 76ms/step - loss: 0.7534 - accuracy: 0.7355 - val_loss: 0.8571 - val_accuracy: 0.7069
    Epoch 9/10
    782/782 [==============================] - 62s 79ms/step - loss: 0.7115 - accuracy: 0.7512 - val_loss: 0.9293 - val_accuracy: 0.6854
    Epoch 10/10
    782/782 [==============================] - 62s 79ms/step - loss: 0.6762 - accuracy: 0.7632 - val_loss: 0.8596 - val_accuracy: 0.7078
    CNN Error: 29.22%
    Accuracy: 70.78%
```

```
[ ] loss, accuracy = model.evaluate(X_test, y_test)
    print(loss, accuracy*100,"%")
```

```
    313/313 [==============================] - 4s 11ms/step - loss: 0.8596 - accuracy: 0.7078
    0.8595006684684753 70.77999711036682 %
```

Creating a dense connected ANN - 21BAI1007

```
def model2():
    model.add(Dense(64, activation='relu', input_shape=(32, 32, 3)))
    model.add(Dropout(0.2))
    model.add(Dense(32, activation='relu'))
    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model2 = model2()
model2.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, verbose=1)
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("Accuracy: %.2f%%" % (scores[1]*100))
```

21BAI1007_CNNvsDense.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:36 AM

Comment Share

+ Code + Text

Connect T4 Colab AI

```
782/782 [==============================] - 59s 75ms/step - loss: 0.7108 - accuracy: 0.7809 - val_loss: 0.9402 - val_accuracy: 0.7165
Epoch 7/10
782/782 [==============================] - 60s 76ms/step - loss: 0.6721 - accuracy: 0.7913 - val_loss: 0.9638 - val_accuracy: 0.7129
Epoch 8/10
782/782 [==============================] - 56s 72ms/step - loss: 0.6434 - accuracy: 0.8000 - val_loss: 0.9777 - val_accuracy: 0.7131
Epoch 9/10
782/782 [==============================] - 59s 76ms/step - loss: 0.6191 - accuracy: 0.8070 - val_loss: 0.9453 - val_accuracy: 0.7129
Epoch 10/10
782/782 [==============================] - 57s 73ms/step - loss: 0.5968 - accuracy: 0.8143 - val_loss: 0.9574 - val_accuracy: 0.7196
CNN Error: 28.04%
Accuracy: 71.96%
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(loss, accuracy*100,"%")
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.9574 - accuracy: 0.7196
0.9574005007743835 71.9600021839141 8 %
```

Results

For CNN Network:

– Accuracy: 70.77%

For Dense Network

– Accuracy: 71.96%