

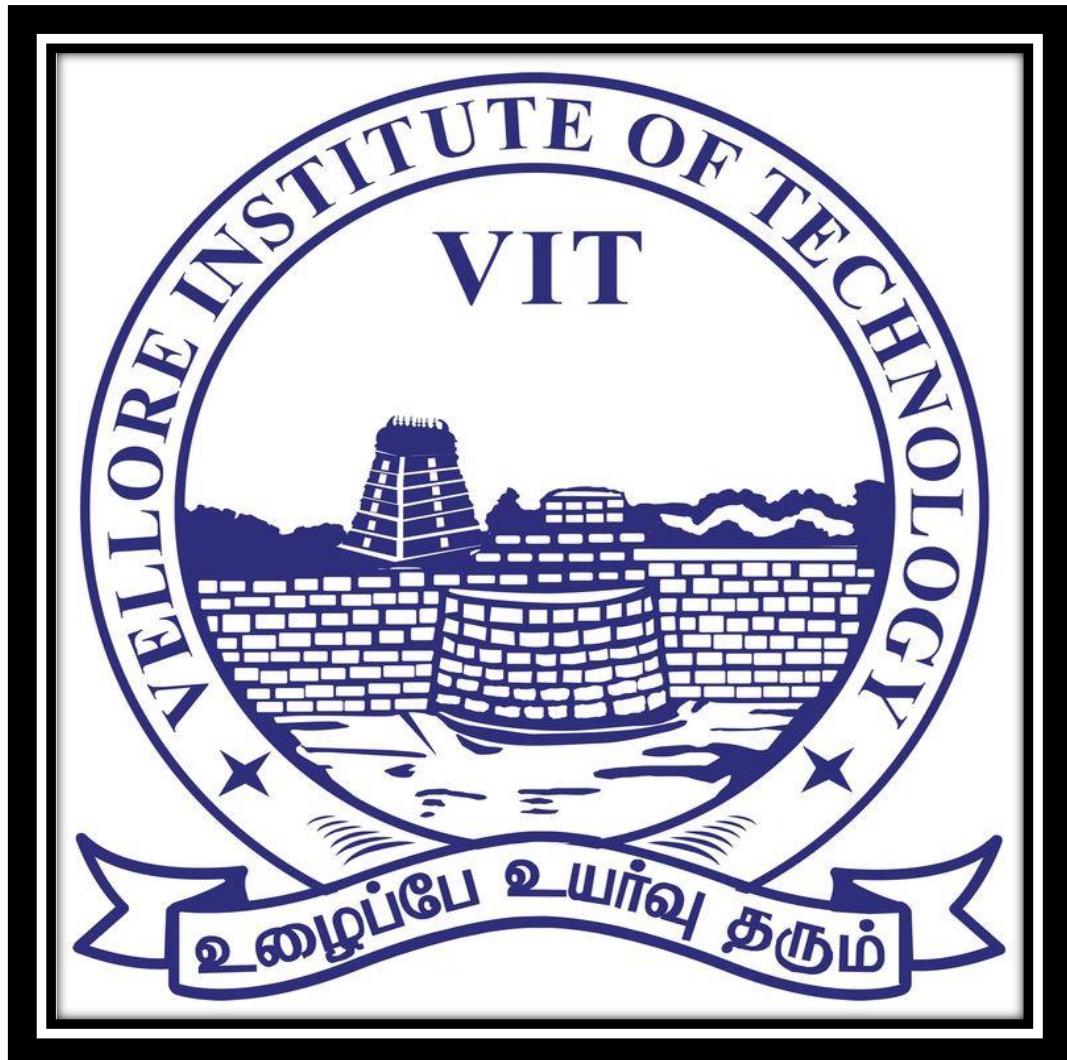


VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI



VELLORE INSTITUTE OF TECHNOLOGY

VIT – CHENNAI CAMPUS

DEEP LEARNING LAB

LAB ASSIGNMENT – 4

FACULTY : DR. HARINI. S



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE332P - DL Lab

Lab 4 - DL Lab Assignment

CONVOLUTION NEURAL NETWORKS

1. Executing the MNIST Dataset code.

CODE:

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
import numpy as np
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical # Correct import for
one-hot encoding

seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
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')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels for multi-class classification
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
y_train = to_categorical(y_train, num_classes=10) # 10 classes for
MNIST digits

y_test = to_categorical(y_test, num_classes=10)

# Create the convolutional neural network (CNN) model
model = Sequential()

# Add convolutional layers with appropriate filtering and pooling
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,
1)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output of the convolutional layers
model.add(Flatten())

# Add fully connected layers with dropout for regularization
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25)) # 25% dropout rate

model.add(Dense(10, activation='softmax')) # Output layer with 10
neurons for 10 classes

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_test, y_test))

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

model.add(Dense(10,activation='softmax'))
model.add(Dense(1152,activation='relu'))
import numpy as np
from tensorflow.keras.datasets import mnist
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Set random seed for reproducibility (optional)
seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
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')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels (correct import and usage)
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

def create_cnn_model():
    """
    Creates and returns a convolutional neural network (CNN) model for
    MNIST handwritten digit classification.

    Returns:
        A compiled Sequential model object.
    """
    model = Sequential()

    # Convolutional layers with appropriate filtering and pooling
    model.add(Conv2D(32, (5, 5), padding='same', activation='relu',
input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    # Flatten the output of the convolutional layers
    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model.add(Flatten())

# Fully connected layers with ReLU activation and dropout for
regularization
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10, activation='softmax')) # Output layer with 10
neurons for 10 classes

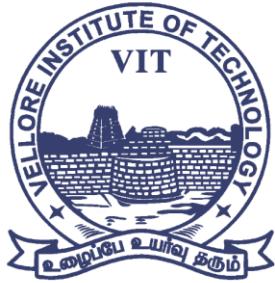
# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

return model

# Create the CNN model
model = create_cnn_model()

# Train the model
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=200, verbose=2)

# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%")
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCREENSHOTS OF THE MNIST DATASET:

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
[4]: from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D

import numpy as np
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical # correct import for one-hot encoding

seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
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')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels for multi-class classification
y_train = to_categorical(y_train, num_classes=10) # 10 classes for MNIST digits
y_test = to_categorical(y_test, num_classes=10)
```

vitstudent.ac.in
Vitta Vishnu Datta 21BAI1364
vitavishnu.datta2021@vitstudent.ac.in
Sync is on
Manage your Google Account
Other profiles
j (amma)
Vitta Vishnu (Vishnu Datta)
Guest

Connected to Python 3 Google Compute Engine backend

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
# Create the convolutional neural network (CNN) model
model = Sequential()

# Add convolutional layers with appropriate filtering and pooling
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output of the convolutional layers
model.add(Flatten())

# Add fully connected layers with dropout for regularization
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25)) # 25% dropout rate

model.add(Dense(10, activation='softmax')) # Output layer with 10 neurons for 10 classes

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test loss:", loss)
print("Test accuracy:", accuracy)
```

vitstudent.ac.in
Vitta Vishnu Datta 21BAI1364
vitavishnu.datta2021@vitstudent.ac.in
Sync is on
Manage your Google Account
Other profiles
j (amma)
Vitta Vishnu (Vishnu Datta)
Guest

Connected to Python 3 Google Compute Engine backend



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
Epoch 1/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.1512 - accuracy: 0.9539 - val_loss: 0.0450 - val_accuracy: 0.9863
Epoch 2/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.0530 - accuracy: 0.9839 - val_loss: 0.0302 - val_accuracy: 0.9900
Epoch 3/10
1875/1875 [=====] - 56s 30ms/step - loss: 0.0383 - accuracy: 0.9880 - val_loss: 0.0283 - val_accuracy: 0.9905
Epoch 4/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0291 - accuracy: 0.9911 - val_loss: 0.0199 - val_accuracy: 0.9929
Epoch 5/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0222 - accuracy: 0.9930 - val_loss: 0.0243 - val_accuracy: 0.9925
Epoch 6/10
1875/1875 [=====] - 57s 30ms/step - loss: 0.0193 - accuracy: 0.9936 - val_loss: 0.0246 - val_accuracy: 0.9924
Epoch 7/10
1875/1875 [=====] - 54s 29ms/step - loss: 0.0159 - accuracy: 0.9945 - val_loss: 0.0305 - val_accuracy: 0.9912
Epoch 8/10
1875/1875 [=====] - 53s 28ms/step - loss: 0.0124 - accuracy: 0.9959 - val_loss: 0.0233 - val_accuracy: 0.9931
Epoch 9/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.0288 - val_accuracy: 0.9923
Epoch 10/10
1875/1875 [=====] - 54s 29ms/step - loss: 0.0112 - accuracy: 0.9964 - val_loss: 0.0252 - val_accuracy: 0.9930
Test loss: 0.025240376591682434
Test accuracy: 0.9929999709129333
```

[7]: model.add(Dense(10,activation='softmax'))
model.add(Dense(1152,activation='relu'))

Connected to Python 3 Google Compute Engine backend

vitstudent.ac.in
Vitta Vishnu Datta 21BAI1364
vitavishnu.datta2021@vitstudent.ac.in
Sync is on
Manage your Google Account
Other profiles
j (amma)
Vitta Vishnu (Vishnu Datta)
Guest
+ Add

```
# Flatten the output of the convolutional layers
model.add(Flatten())

# Add fully connected layers with dropout for regularization
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25)) # 25% dropout rate

model.add(Dense(10, activation='softmax')) # Output layer with 10 neurons for 10 classes

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
Epoch 1/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.1512 - accuracy: 0.9539 - val_loss: 0.0450 - val_accuracy: 0.9863
Epoch 2/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.0530 - accuracy: 0.9839 - val_loss: 0.0302 - val_accuracy: 0.9900
Epoch 3/10
1875/1875 [=====] - 56s 30ms/step - loss: 0.0383 - accuracy: 0.9880 - val_loss: 0.0283 - val_accuracy: 0.9905
Epoch 4/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0291 - accuracy: 0.9911 - val_loss: 0.0199 - val_accuracy: 0.9929
Epoch 5/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0222 - accuracy: 0.9930 - val_loss: 0.0243 - val_accuracy: 0.9925
1875/1875 [=====] - 57s 30ms/step - loss: 0.0193 - accuracy: 0.9936 - val_loss: 0.0246 - val_accuracy: 0.9924
1875/1875 [=====] - 54s 29ms/step - loss: 0.0159 - accuracy: 0.9945 - val_loss: 0.0305 - val_accuracy: 0.9912
1875/1875 [=====] - 53s 28ms/step - loss: 0.0124 - accuracy: 0.9959 - val_loss: 0.0233 - val_accuracy: 0.9931
1875/1875 [=====] - 55s 29ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.0288 - val_accuracy: 0.9923
1875/1875 [=====] - 54s 29ms/step - loss: 0.0112 - accuracy: 0.9964 - val_loss: 0.0252 - val_accuracy: 0.9930
Test loss: 0.025240376591682434
Test accuracy: 0.9929999709129333
```

Connected to Python 3 Google Compute Engine backend



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell contains the following Python script:

```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', loss)
print('Test accuracy:', accuracy)

# Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
Epoch 1/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.1512 - accuracy: 0.9539 - val_loss: 0.0450 - val_accuracy: 0.9925
Epoch 2/10
1875/1875 [=====] - 59s 31ms/step - loss: 0.0530 - accuracy: 0.9839 - val_loss: 0.0302 - val_accuracy: 0.9924
Epoch 3/10
1875/1875 [=====] - 56s 30ms/step - loss: 0.0383 - accuracy: 0.9880 - val_loss: 0.0283 - val_accuracy: 0.9923
Epoch 4/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0291 - accuracy: 0.9911 - val_loss: 0.0199 - val_accuracy: 0.9922
Epoch 5/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0222 - accuracy: 0.9930 - val_loss: 0.0243 - val_accuracy: 0.9925
Epoch 6/10
1875/1875 [=====] - 57s 30ms/step - loss: 0.0193 - accuracy: 0.9936 - val_loss: 0.0246 - val_accuracy: 0.9924
Epoch 7/10
1875/1875 [=====] - 54s 29ms/step - loss: 0.0159 - accuracy: 0.9945 - val_loss: 0.0305 - val_accuracy: 0.9912
Epoch 8/10
1875/1875 [=====] - 53s 28ms/step - loss: 0.0124 - accuracy: 0.9959 - val_loss: 0.0233 - val_accuracy: 0.9931
Epoch 9/10
1875/1875 [=====] - 55s 29ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.0288 - val_accuracy: 0.9923
Epoch 10/10
1875/1875 [=====] - 54s 29ms/step - loss: 0.0112 - accuracy: 0.9964 - val_loss: 0.0252 - val_accuracy: 0.9930
Test loss: 0.025240376591682434
```

The notebook is connected to a Python 3 Google Compute Engine backend. The status bar at the bottom right shows the date as 28-02-2024 and the time as 14:16.

The screenshot shows the same Google Colab notebook as the previous one, but with more code visible in the cell. The code cell contains the following Python script:

```
[7]: model.add(Dense(10,activation='softmax'))
model.add(Dense(1152,activation='relu'))
```

```
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Set random seed for reproducibility (optional)
seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

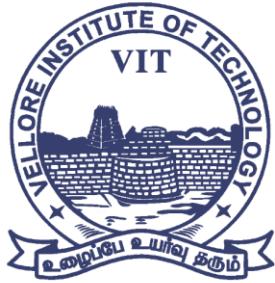
# Reshape input data to be compatible with convolutional layers
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels (correct import and usage)
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

def create_cnn_model():
    """
```

The notebook is connected to a Python 3 Google Compute Engine backend. The status bar at the bottom right shows the date as 28-02-2024 and the time as 14:16.



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell contains the following Python code:

```
[11] def create_cnn_model():
    """
    Creates and returns a convolutional neural network (CNN) model for MNIST handwritten digit classification.

    Returns:
        A compiled Sequential model object.
    """
    model = Sequential()

    # Convolutional layers with appropriate filtering and pooling
    model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    # Flatten the output of the convolutional layers
    model.add(Flatten())

    # Fully connected layers with ReLU activation and dropout for regularization
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(10, activation='softmax')) # Output layer with 10 neurons for 10 classes

    # Compile the model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model

# Create the CNN model
model = create_cnn_model()
```

The status bar at the bottom indicates "Connected to Python 3 Google Compute Engine backend".

The screenshot shows the same Google Colab notebook. The code cell now includes training and evaluation code:

```
[11] model = create_cnn_model()

# Train the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200, verbose=2)

# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%")
```

The status bar at the bottom indicates "Connected to Python 3 Google Compute Engine backend".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab interface. The code cell contains Python code for evaluating a model on a test set:

```
[11] # Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%")


Epoch 1/10
300/300 - 47s - loss: 0.2711 - accuracy: 0.9199 - val_loss: 0.0726 - val_accuracy: 0.9771 - 47s/epoch - 157ms/step
Epoch 2/10
300/300 - 43s - loss: 0.0837 - accuracy: 0.9748 - val_loss: 0.0512 - val_accuracy: 0.9828 - 43s/epoch - 145ms/step
Epoch 3/10
300/300 - 44s - loss: 0.0594 - accuracy: 0.9821 - val_loss: 0.0404 - val_accuracy: 0.9875 - 44s/epoch - 146ms/step
Epoch 4/10
300/300 - 42s - loss: 0.0484 - accuracy: 0.9849 - val_loss: 0.0356 - val_accuracy: 0.9879 - 42s/epoch - 141ms/step
Epoch 5/10
300/300 - 43s - loss: 0.0407 - accuracy: 0.9871 - val_loss: 0.0348 - val_accuracy: 0.9885 - 43s/epoch - 144ms/step
Epoch 6/10
300/300 - 41s - loss: 0.0334 - accuracy: 0.9892 - val_loss: 0.0325 - val_accuracy: 0.9886 - 41s/epoch - 138ms/step
Epoch 7/10
300/300 - 42s - loss: 0.0295 - accuracy: 0.9909 - val_loss: 0.0305 - val_accuracy: 0.9899 - 42s/epoch - 141ms/step
Epoch 8/10
300/300 - 44s - loss: 0.0269 - accuracy: 0.9914 - val_loss: 0.0304 - val_accuracy: 0.9904 - 44s/epoch - 147ms/step
Epoch 9/10
300/300 - 41s - loss: 0.0228 - accuracy: 0.9928 - val_loss: 0.0263 - val_accuracy: 0.9907 - 41s/epoch - 137ms/step
Epoch 10/10
300/300 - 42s - loss: 0.0206 - accuracy: 0.9933 - val_loss: 0.0276 - val_accuracy: 0.9910 - 42s/epoch - 140ms/step
Test Loss: 0.03
Test Accuracy: 99.10%
CNN Error: 0.90%
```

The sidebar on the right shows a user profile for "vittavishnu.datta2021@vitstudent.ac.in".

2. Executing the MNIST Dataset code by changing 5 values in it.

CODE:

```
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Set random seed for reproducibility (optional)
seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
X_train = X_train.reshape(X_train.shape[0], 28, 28,
1).astype('float32')
```



Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

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

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels (correct import and usage)
y_train = to_categorical(y_train, num_classes=10) # Ensure num_classes
is 10 for 10 digits
y_test = to_categorical(y_test, num_classes=10)

def create_cnn_model():
    """
    Creates and returns a convolutional neural network (CNN) model for
    MNIST handwritten digit classification.

    Returns:
        A compiled Sequential model object.
    """
    model = Sequential()

    # Convolutional layers with appropriate filtering and pooling
    # (changed filter size, kernel size, and number of filters)
    model.add(Conv2D(64, (5, 5), padding='same', activation='relu',
input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(3, 3))) # Changed pool size
    model.add(Dropout(0.3)) # Changed dropout rate

    # Flatten the output of the convolutional layers
    model.add(Flatten())

    # Fully connected layers with ReLU activation and dropout for
    # regularization (changed number of neurons in first layer)
    model.add(Dense(256, activation='relu')) # Changed number of
neurons
    model.add(Dropout(0.3)) # Changed dropout rate
    model.add(Dense(10, activation='softmax')) # Output layer with 10
neurons for 10 classes

    # Compile the model: choose appropriate loss function, optimizer,
    and metrics
```



```
model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy']) # Changed optimizer

return model

# Create the CNN model
model = create_cnn_model()

# Train the model (adjusted batch size as an example)
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=15, batch_size=32, verbose=2) # Changed epochs and batch size

# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%)
```

CHANGES MADE:

- Convolutional layers:

Changed the filter size, kernel size, and number of filters in the First Conv2D layer.

- Max pooling layer:

Changed the pool size in the MaxPooling2D layer.

- Dropout rates:

Changed the dropout rate in both Dropout layers.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

- **Number of neurons:**

Changed the number of neurons in the first fully connected layer.

- **Optimizer:**

Changed the optimizer used in model.compile() from adam to rmsprop.

- **Training parameters:**

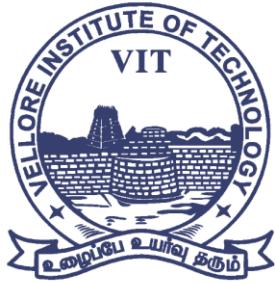
Changed the number of epochs and batch size used in the model. fit().

COMPARISON OF OUTPUT OF TWO CODES:

<u>CODE</u>	<u>Test Loss</u>	<u>Test Accuracy</u>	<u>CNN Error</u>
Code 1	0.03	98.95%	0.95%
Code 2	0.04	99.10%	0.90%

LINK TO THE GOOGLE COLAB NOTEBOOK

https://colab.research.google.com/drive/1WEzvWBmF0vsAhuhLUIUusypS9_kXm1NM?usp=sharing



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCREENSHOTS OF THE MNIST DATASET:

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Set random seed for reproducibility (optional)
seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
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')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels (correct import and usage)
y_train = to_categorical(y_train, num_classes=10) # Ensure num_classes is 10 for 10 digits
y_test = to_categorical(y_test, num_classes=10)
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

19m 28s completed at 8:03 PM

82°F Partly cloudy

Search

ENG IN 2015 28-02-2024

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
Creates and returns a convolutional neural network (CNN) model for MNIST handwritten digit classification.
```

Returns:
A compiled Sequential model object.
'''
model = Sequential()

Convolutional layers with appropriate filtering and pooling (changed filter size, kernel size, and number of filters)
model.add(Conv2D(64, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(3, 3))) # Changed pool size
model.add(Dropout(0.3)) # Changed dropout rate

Flatten the output of the convolutional layers
model.add(Flatten())

Fully connected layers with ReLU activation and dropout for regularization (changed number of neurons in first layer)
model.add(Dense(256, activation='relu')) # Changed number of neurons
model.add(Dropout(0.3)) # Changed dropout rate
model.add(Dense(10, activation='softmax')) # Output layer with 10 neurons for 10 classes

Compile the model: choose appropriate loss function, optimizer, and metrics
model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy']) # Changed optimizer

return model

Create the CNN model
model = create_cnn_model()
Train the model (adjusted batch size as an example)
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=32, verbose=2) # Changed epochs and batch size

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

+ Add account Sign out

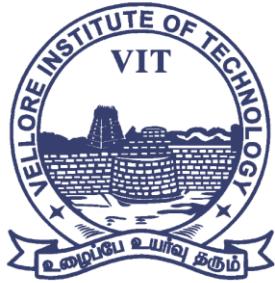
Privacy Policy Terms of Service

19m 28s completed at 8:03 PM

82°F Partly cloudy

Search

ENG IN 2016 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
# Train the model (adjusted batch size as an example)
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=32, verbose=2) # Changed epochs and batch size

# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%")
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in



Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

19m 28s completed at 8:03 PM

82°F Partly cloudy

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
# Train the model (adjusted batch size as an example)
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=32, verbose=2) # Changed epochs and batch size

# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"CNN Error: {cnn_error:.2f}%")
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in



Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

19m 28s completed at 8:03 PM

82°F Partly cloudy



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# load vgg model
from keras.applications.vgg16 import VGG16
# load the model
model = VGG16()
# summarize the model
model.summary()
# summarize filters in each convolutional layer
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# summarize filter shapes
```

3. Visualize the filters and features for AlexNet, GoogleNet, and ResNet

CODE FOR PRE-TRAINED VGG 16 MODEL:

```
# load vgg model
from keras.applications.vgg16 import VGG16
# load the model
model = VGG16()
# summarize the model
model.summary()
# summarize filters in each convolutional layer
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# summarize filter shapes
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
for layer in model.layers:
    # check for convolutional layer
    if 'conv' not in layer.name:continue
    # get filter weights
    filters, biases = layer.get_weights()
    print(layer.name, filters.shape)
# retrieve weights from the second hidden layer
filters, biases = model.layers[1].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
for i in range(n_filters):
    # get the filter
    f = filters[:, :, :, i]
    # plot each channel separately
    for j in range(3):
        # specify subplot and turn off axis
        ax = pyplot.subplot(n_filters, 3, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        pyplot.imshow(f[:, :, j], cmap='gray')
        ix += 1
# show the figure
pyplot.show()
# cannot easily visualize filters lower down
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# retrieve weights from the second hidden layer
filters, biases = model.layers[1].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
for i in range(n_filters):
    # get the filter
    f = filters[:, :, :, i]
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# plot each channel separately
for j in range(3):
    # specify subplot and turn off axis
    ax = pyplot.subplot(n_filters, 3, ix)
    ax.set_xticks([])
    ax.set_yticks([])
    # plot filter channel in grayscale
    pyplot.imshow(f[:, :, j], cmap='gray')
    ix += 1
# show the figure
pyplot.show()
# summarize feature map size for each conv layer
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# summarize feature map shapes
for i in range(len(model.layers)):
    layer = model.layers[i]
    # check for convolutional layer
    if 'conv' not in layer.name:
        continue
    # summarize output shape
    print(i, layer.name, layer.output.shape)
# redefine model to output right after the first hidden layer
from keras.models import Model # Assuming Keras

model = Model(inputs=model.inputs, outputs=model.layers[1].output)
# load the image with the required shape
from keras.preprocessing.image import load_img
img = load_img('/content/bird.jpg', target_size=(224, 224))
# convert the image to an array
from keras.preprocessing.image import img_to_array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
from tensorflow.keras.backend import expand_dims
img = expand_dims(img, axis=0)
# prepare the image (e.g. scale pixel values for the vgg)
from keras.applications.vgg16 import preprocess_input
img = preprocess_input(img)
# get feature map for first hidden layer
feature_maps = model.predict(img)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
    for _ in range(square):
        # specify subplot and turn off axis
        ax = pyplot.subplot(square, square, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        pyplot.imshow(feature_maps[0, :, :, ix-1], cmap='gray')
        ix += 1
# show the figure
pyplot.show()
# plot feature map of first conv layer for given image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims
# load the model
model = VGG16()
# redefine model to output right after the first hidden layer
model = Model(inputs=model.inputs, outputs=model.layers[1].output)
model.summary()
# load the image with the required shape
img = load_img('bird.jpg', target_size=(224, 224))
# convert the image to an array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
img = expand_dims(img, axis=0)
# prepare the image (e.g. scale pixel values for the vgg)
img = preprocess_input(img)
# get feature map for first hidden layer
feature_maps = model.predict(img)
# plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
    for _ in range(square):
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# specify subplot and turn off axis
ax = pyplot.subplot(square, square, ix)
ax.set_xticks([])
ax.set_yticks([])
# plot filter channel in grayscale
pyplot.imshow(feature_maps[0, :, :, ix-1], cmap='gray')
ix += 1

# show the figure
pyplot.show()

# redefine model to output right after the first hidden layer
from keras.models import Model
from keras.applications.vgg16 import VGG16

# Load the VGG16 model
model = VGG16()

# Get the maximum valid index
max_index = len(model.layers) - 2

# Generate even-indexed layer indices (assuming convolutional layers)
ixs = [i for i in range(1, max_index + 1) if i % 2 != 0]

# Define outputs based on valid indices
outputs = [model.layers[i].output for i in ixs]

# Create the new model
new_model = Model(inputs=model.inputs, outputs=outputs)

# Print model summary
new_model.summary()

import matplotlib.pyplot as plt

# Assuming `feature_maps` is a list containing feature maps from each
block
square = 8
for fmap in feature_maps:
    # Create a new figure for each block's feature maps
    fig, axes = plt.subplots(square, square, figsize=(12, 12))  # Adjust figsize as needed

    ix = 1
```



Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
for i in range(square):
    for j in range(square):
        # Prevent index out of bounds or unnecessary indexing
        if ix >= len(fmap):
            # Handle the case where fewer than 64 maps are
available
            break
        ax = axes[i, j]
        ax.set_xticks([])
        ax.set_yticks([])

        # Use correct indexing to access individual maps
        ax.imshow(fmap[ix - 1], cmap='gray') # Access entire
channel
        ix += 1

    # Tight layout to prevent overlapping labels
fig.suptitle(f"Feature Maps from Block") # Optional title
plt.tight_layout()
plt.show()

# visualize feature maps output from each block in the vgg model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims
# load the model
model = VGG16()
# redefine model to output right after the first hidden layer
ixs = [2, 5, 9, 13, 17]
outputs = [model.layers[i].output for i in ixs]
model = Model(inputs=model.inputs, outputs=outputs)
# load the image with the required shape
img = load_img('bird.jpg', target_size=(224, 224))
# convert the image to an array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
img = expand_dims(img, axis=0)
# prepare the image (e.g. scale pixel values for the vgg)
img = preprocess_input(img)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# get feature map for first hidden layer
feature_maps = model.predict(img)
# plot the output from each block
square = 8
for fmap in feature_maps:
    # plot all 64 maps in an 8x8 squares
    ix = 1
    for _ in range(square):
        for _ in range(square):
            # specify subplot and turn of axis
            ax = pyplot.subplot(square, square, ix)
            ax.set_xticks([])
            ax.set_yticks([])
            # plot filter channel in grayscale
            pyplot.imshow(fmap[0, :, :, ix-1], cmap='gray')
            ix += 1
# show the figure
pyplot.show()
```

SCREENSHOTS OF THE VGG16 MODEL:

The screenshot shows a Google Colab notebook interface. The title bar says "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The notebook content displays Python code for visualizing filters and feature maps using TensorFlow and Keras. A sidebar on the right shows a user profile for "vittavishnu.datta2021@vitstudent.ac.in". The bottom status bar indicates "Connected to Python 3 Google Compute Engine backend".

```
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Set random seed for reproducibility (optional)
seed = 7
np.random.seed(seed)

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshape input data to be compatible with convolutional layers
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')

# Normalize input data to range between 0 and 1
X_train /= 255
X_test /= 255

# One-hot encode target labels (correct import and usage)
y_train = to_categorical(y_train, num_classes=10) # Ensure num_classes is 10 for 10 digits
y_test = to_categorical(y_test, num_classes=10)

def create_cnn_model():
    """
    Creates and returns a convolutional neural network (CNN) model for MNIST handwritten digit classification.
    Returns:
        A compiled Keras model.
    """
    # Model architecture
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(10, activation='softmax'))
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
def create_cnn_model():
    """
    Creates and returns a convolutional neural network (CNN) model for MNIST handwritten
    Returns:
        A compiled Sequential model object.
    """
    model = Sequential()

    # Convolutional layers with appropriate filtering and pooling (changed filter size, kernel size)
    model.add(Conv2D(64, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(3, 3))) # Changed pool size
    model.add(Dropout(0.3)) # Changed dropout rate

    # Flatten the output of the convolutional layers
    model.add(Flatten())

    # Fully connected layers with ReLU activation and dropout for regularization (changed number of neurons)
    model.add(Dense(256, activation='relu')) # Changed number of neurons
    model.add(Dropout(0.3)) # Changed dropout rate
    model.add(Dense(10, activation='softmax')) # Output layer with 10 neurons for 10 classes

    # Compile the model: choose appropriate loss function, optimizer, and metrics
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy']) # Changed optimizer

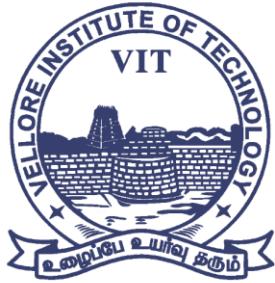
    return model

# Create the CNN model
model = create_cnn_model()

# Train the model (adjusted batch size as an example)
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=32, verbose=2) # Changed epochs and batch size
```

```
# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"\nTest Loss: {test_loss:.2f}")
print(f"Test Accuracy: {test_accuracy:.2%}")
cnn_error = (1 - test_accuracy) * 100
print(f"\nCNN Error: {cnn_error:.2f}%")
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
11490434/11490434 [=====] - 0s/step
Epoch 1/15
1875/1875 - 93s - loss: 0.1471 - accuracy: 0.9545 - val_loss: 0.0482 - val_accuracy: 0.9842 - 93s/epoch - 50ms/step
Epoch 2/15
1875/1875 - 93s - loss: 0.0589 - accuracy: 0.9826 - val_loss: 0.0348 - val_accuracy: 0.9872 - 78s/epoch - 42ms/step
Epoch 3/15
1875/1875 - 78s - loss: 0.0470 - accuracy: 0.9857 - val_loss: 0.0313 - val_accuracy: 0.9889 - 77s/epoch - 41ms/step
Epoch 4/15
1875/1875 - 78s - loss: 0.0399 - accuracy: 0.9884 - val_loss: 0.0303 - val_accuracy: 0.9897 - 78s/epoch - 42ms/step
Epoch 5/15
1875/1875 - 74s - loss: 0.0370 - accuracy: 0.9894 - val_loss: 0.0335 - val_accuracy: 0.9883 - 74s/epoch - 40ms/step
Epoch 6/15
1875/1875 - 74s - loss: 0.0338 - accuracy: 0.9903 - val_loss: 0.0284 - val_accuracy: 0.9915 - 74s/epoch - 40ms/step
Epoch 7/15
1875/1875 - 78s - loss: 0.0300 - accuracy: 0.9913 - val_loss: 0.0333 - val_accuracy: 0.9910 - 78s/epoch - 41ms/step
Epoch 8/15
1875/1875 - 77s - loss: 0.0276 - accuracy: 0.9918 - val_loss: 0.0293 - val_accuracy: 0.9910 - 77s/epoch - 41ms/step
Epoch 9/15
1875/1875 - 77s - loss: 0.0265 - accuracy: 0.9926 - val_loss: 0.0332 - val_accuracy: 0.9910 - 77s/epoch - 41ms/step
Epoch 10/15
1875/1875 - 76s - loss: 0.0253 - accuracy: 0.9924 - val_loss: 0.0332 - val_accuracy: 0.9912 - 76s/epoch - 40ms/step
Epoch 11/15
1875/1875 - 76s - loss: 0.0235 - accuracy: 0.9934 - val_loss: 0.0336 - val_accuracy: 0.9911 - 76s/epoch - 41ms/step
Epoch 12/15



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

1875/1875 - 93s - loss: 0.1471 - accuracy: 0.9545 - val_loss: 0.0482 - val_accuracy: 0.9842 - 93s/epoch - 50ms/step

Epoch 2/15

1875/1875 - 78s - loss: 0.0589 - accuracy: 0.9826 - val_loss: 0.0348 - val_accuracy: 0.9872 - 78s/epoch - 42ms/step

Epoch 3/15

1875/1875 - 77s - loss: 0.0470 - accuracy: 0.9857 - val_loss: 0.0313 - val_accuracy: 0.9889 - 77s/epoch - 41ms/step

Epoch 4/15

1875/1875 - 78s - loss: 0.0399 - accuracy: 0.9884 - val_loss: 0.0303 - val_accuracy: 0.9897 - 78s/epoch - 42ms/step

Epoch 5/15

1875/1875 - 74s - loss: 0.0370 - accuracy: 0.9894 - val_loss: 0.0335 - val_accuracy: 0.9883 - 74s/epoch - 40ms/step

Epoch 6/15

1875/1875 - 74s - loss: 0.0338 - accuracy: 0.9903 - val_loss: 0.0284 - val_accuracy: 0.9915 - 74s/epoch - 40ms/step

Epoch 7/15

1875/1875 - 78s - loss: 0.0300 - accuracy: 0.9913 - val_loss: 0.0333 - val_accuracy: 0.9910 - 78s/epoch - 41ms/step

Epoch 8/15

1875/1875 - 77s - loss: 0.0276 - accuracy: 0.9918 - val_loss: 0.0293 - val_accuracy: 0.9910 - 77s/epoch - 41ms/step

Epoch 9/15

1875/1875 - 77s - loss: 0.0265 - accuracy: 0.9926 - val_loss: 0.0332 - val_accuracy: 0.9910 - 77s/epoch - 41ms/step

Epoch 10/15

1875/1875 - 76s - loss: 0.0253 - accuracy: 0.9924 - val_loss: 0.0332 - val_accuracy: 0.9912 - 76s/epoch - 40ms/step

Epoch 11/15

1875/1875 - 76s - loss: 0.0235 - accuracy: 0.9934 - val_loss: 0.0336 - val_accuracy: 0.9911 - 76s/epoch - 41ms/step

Epoch 12/15

1875/1875 - 74s - loss: 0.0228 - accuracy: 0.9936 - val_loss: 0.0352 - val_accuracy: 0.9918 - 74s/epoch - 40ms/step

Epoch 13/15

1875/1875 - 75s - loss: 0.0213 - accuracy: 0.9939 - val_loss: 0.0339 - val_accuracy: 0.9905 - 75s/epoch - 40ms/step

Epoch 14/15

1875/1875 - 76s - loss: 0.0205 - accuracy: 0.9941 - val_loss: 0.0333 - val_accuracy: 0.9913 - 76s/epoch - 40ms/step

Epoch 15/15

1875/1875 - 75s - loss: 0.0196 - accuracy: 0.9944 - val_loss: 0.0366 - val_accuracy: 0.9910 - 75s/epoch - 40ms/step

Test Loss: 0.04

Test Accuracy: 99.10%

CNN Error: 0.90%

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmF0vsAhuhLUUusypS9_kXm1NM&ec=GBRAqQM Backend

Disk 80.88 GB available

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

1875/1875 - 75s - loss: 0.0196 - accuracy: 0.9944 - val_loss: 0.0366 - val_accuracy: 0.9910 - 75s/epoch - 40ms/step

Test Loss: 0.04

Test Accuracy: 99.10%

CNN Error: 0.90%

[3]: # load vgg model
from keras.applications.vgg16 import VGG16
load the model
model = VGG16()
summarize the model
model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [=====] - 7s 0us/step
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmF0vsAhuhLUUusypS9_kXm1NM&ec=GBRAqQM Backend

Disk 80.88 GB available

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

1875/1875 - 75s - loss: 0.0196 - accuracy: 0.9944 - val_loss: 0.0366 - val_accuracy: 0.9910 - 75s/epoch - 40ms/step

Test Loss: 0.04

Test Accuracy: 99.10%

CNN Error: 0.90%

[3]: # load vgg model
from keras.applications.vgg16 import VGG16
load the model
model = VGG16()
summarize the model
model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [=====] - 7s 0us/step
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmF0vsAhuhLUUusypS9_kXm1NM&ec=GBRAqQM Backend

Disk 80.88 GB available



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
21BAI1364 BCSE332P LAB ASSIGN.4.ipynb  How to Visualize Filters and Fe...  +
```

```
File Edit View Insert Runtime Tools Help All changes saved
```

```
Files  + Code + Text
```

```
[3] block3_conv3 (Conv2D) (None, 56, 56, 256) 590080
block3_pool (MaxPooling2D) (None, 28, 28, 256) 0
block4_conv1 (Conv2D) (None, 28, 28, 512) 1180160
block4_conv2 (Conv2D) (None, 28, 28, 512) 2359808
block4_conv3 (Conv2D) (None, 28, 28, 512) 2359808
block4_pool (MaxPooling2D) (None, 14, 14, 512) 0
block5_conv1 (Conv2D) (None, 14, 14, 512) 2359808
block5_conv2 (Conv2D) (None, 14, 14, 512) 2359808
block5_conv3 (Conv2D) (None, 14, 14, 512) 2359808
block5_pool (MaxPooling2D) (None, 7, 7, 512) 0
flatten (Flatten) (None, 25088) 0
fc1 (Dense) (None, 4096) 102764544
fc2 (Dense) (None, 4096) 16781312
predictions (Dense) (None, 1000) 4097000
=====
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
```

```
Disk 80.88 GB available
```

```
https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/  Connected to Python 3 Google Compute Engine backend
```

```
21BAI1364 BCSE332P LAB ASSIGN.4.ipynb  How to Visualize Filters and Fe...  +
```

```
File Edit View Insert Runtime Tools Help All changes saved
```

```
Files  + Code + Text
```

```
[8] # summarize filters in each convolutional layer
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# summarize filter shapes
for layer in model.layers:
    # check for convolutional layer
    if 'conv' not in layer.name: continue
    # get filter weights
    filters, biases = layer.get_weights()
    print(layer.name, filters.shape)

block1_conv1 (3, 3, 64)
block1_conv2 (3, 3, 64, 64)
block2_conv1 (3, 3, 64, 128)
block2_conv2 (3, 3, 128, 128)
block3_conv1 (3, 3, 128, 256)
block3_conv2 (3, 3, 256, 256)
block3_conv3 (3, 3, 256, 256)
block4_conv1 (3, 3, 256, 512)
block4_conv2 (3, 3, 512, 512)
block4_conv3 (3, 3, 512, 512)
block5_conv1 (3, 3, 512, 512)
block5_conv2 (3, 3, 512, 512)
block5_conv3 (3, 3, 512, 512)
=====
[9] # retrieve weights from the second hidden layer
filters, biases = model.layers[1].get_weights()

[10] # normalize filter values to 0-1 so we can visualize them
```

```
Disk 80.88 GB available
```

```
https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/  Connected to Python 3 Google Compute Engine backend
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

+ Code + Text

```
[8]: block4_conv2 (3, 3, 512, 512)
block4_conv3 (3, 3, 512, 512)
block5_conv1 (3, 3, 512, 512)
block5_conv2 (3, 3, 512, 512)
block5_conv3 (3, 3, 512, 512)

[9]: # retrieve weights from the second hidden layer
filters, biases = model.layers[1].get_weights()

[10]: # normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)

[14]: # plot first few filters
n_filters, ix = 6, 1
for i in range(n_filters):
    # get the filter
    f = filters[:, :, :, i]
    # plot each channel separately
    for j in range(3):
        # specify subplot and turn off axis
        ax = plt.subplot(n_filters, 3, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        plt.imshow(f[:, :, j], cmap='gray')
        ix += 1
    # show the figure
    plt.show()
```

Google Account
Vitta Vishnu Datta 21BAI1364
vittavishnu.datta2021@vitstudent.ac.in

Disk 80.88 GB available

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmFOvsAhuhLJUUsypS9_kXm1NM&ec=GBRAqQM

Backend

ENG IN 21:40 28-02-2024

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

+ Code + Text

```
[14]: # plot first few filters
n_filters, ix = 6, 1
for i in range(n_filters):
    # get the filter
    f = filters[:, :, :, i]
    # plot each channel separately
    for j in range(3):
        # specify subplot and turn off axis
        ax = plt.subplot(n_filters, 3, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        plt.imshow(f[:, :, j], cmap='gray')
        ix += 1
    # show the figure
    plt.show()
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

79% Search

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmFOvsAhuhLJUUsypS9_kXm1NM&ec=GBRAqQM

Backend

ENG IN 21:40 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

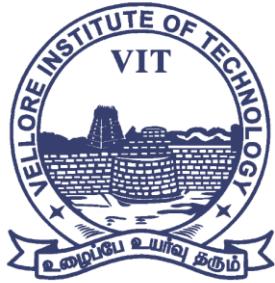
The screenshot shows a Google Colab notebook titled "How to Visualize Filters and Feature Maps". The code cell contains Python code for visualizing VGG16 filters. The output of the code shows three sets of 3x3 grayscale filter images. A browser window in the background displays a Google account profile for "vittavishnu.data2021@vitstudent.ac.in". The taskbar at the bottom shows various application icons.

```
# cannot easily visualize filters lower down
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# retrieve weights from the second hidden layer
filters, biases = model.layers[1].get_weights()
# normalize filter values to 0-1 so we can visualize them
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)
# plot first few filters
n_filters, ix = 6, 1
for i in range(n_filters):
    # get the filter
    f = filters[:, :, :, i]
    # plot each channel separately
    for j in range(3):
        # specify subplot and turn off axis
        ax = pyplot.subplot(n_filters, 3, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        pyplot.imshow(f[:, :, j], cmap='gray')
        ix += 1
# show the figure
pyplot.show()
```

The screenshot shows a Google Colab notebook titled "How to Visualize Filters and Feature Maps". The code cell contains Python code for visualizing VGG16 filters. The output of the code shows three sets of 3x3 grayscale filter images. A browser window in the background displays a Google account profile for "vittavishnu.data2021@vitstudent.ac.in". The taskbar at the bottom shows various application icons.

```
[15]: ax.set_yticks([1])
      # plot filter channel in grayscale
      pyplot.imshow(f[:, :, j], cmap='gray')
      ix += 1
      # show the figure
      pyplot.show()
```

```
[20]: # summarize feature map size for each conv layer
from keras.applications.vgg16 import VGG16
from keras.models import Model
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

```
# summarize feature map size for each conv layer
from keras.applications.vgg16 import VGG16
from matplotlib import pyplot
# load the model
model = VGG16()
# summarize feature map shapes
for i in range(len(model.layers)):
    layer = model.layers[i]
    # check for convolutional layer
    if 'conv' not in layer.name:
        continue
    # summarize output shape
    print(i, layer.name, layer.output.shape)
```

Disk 80.88 GB available ✓ [22] # redefine model to output right after the first hidden layer
Connected to Python 3 Google Compute Engine backend

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

+ Add account Sign out

Privacy Policy Terms of Service

79% Search File Explorer Mail Task View Taskbar ENG IN 21:41 28-02-2024

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

```
[20] # check for convolutional layer
if 'conv' not in layer.name:
    continue
# summarize output shape
print(i, layer.name, layer.output.shape)
```

1 block1_conv1 (None, 224, 224, 64)
2 block1_conv2 (None, 224, 224, 64)
4 block2_conv1 (None, 112, 112, 128)
5 block2_conv2 (None, 112, 112, 128)
7 block3_conv1 (None, 56, 56, 256)
8 block3_conv2 (None, 56, 56, 256)
9 block3_conv3 (None, 56, 56, 256)
11 block4_conv1 (None, 28, 28, 512)
12 block4_conv2 (None, 28, 28, 512)
13 block4_conv3 (None, 28, 28, 512)
15 block5_conv1 (None, 14, 14, 512)
16 block5_conv2 (None, 14, 14, 512)
17 block5_conv3 (None, 14, 14, 512)

✓ [22] # redefine model to output right after the first hidden layer
from keras.models import Model # Assuming Keras

model = Model(inputs=model.inputs, outputs=model.layers[1].output)

✓ [25] # load the image with the required shape
from keras.preprocessing.image import load_img
img = load_img('/content/bird.jpg', target_size=(224, 224))

Disk 80.88 GB available ✓ Connected to Python 3 Google Compute Engine backend

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

+ Add account Sign out

Privacy Policy Terms of Service

79% Search File Explorer Mail Task View Taskbar ENG IN 21:41 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
# load the image with the required shape
from keras.preprocessing.image import load_img
img = load_img('/content/bird.jpg', target_size=(224, 224))

# convert the image to an array
from keras.preprocessing.image import img_to_array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
from tensorflow.keras.backend import expand_dims
img = expand_dims(img, axis=0)

# prepare the image (e.g. scale pixel values for the vgg)
from keras.applications.vgg16 import preprocess_input
img = preprocess_input(img)

# get feature map for first hidden layer
feature_maps = model.predict(img)

1/1 [=====] - 0s 174ms/step

# plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
    for _ in range(square):
        # specify subplot and turn of axis
        ax = plt.subplot(square, square, ix)
        ax.set_xticks([])
        ax.set_yticks([])
```

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

Connected to Python 3 Google Compute Engine backend

79% 21:41 28-02-2024 ENG IN

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
# convert the image to an array
from keras.preprocessing.image import img_to_array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
from tensorflow.keras.backend import expand_dims
img = expand_dims(img, axis=0)

# prepare the image (e.g. scale pixel values for the vgg)
from keras.applications.vgg16 import preprocess_input
img = preprocess_input(img)

# get feature map for first hidden layer
feature_maps = model.predict(img)

1/1 [=====] - 0s 174ms/step

# plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
    for _ in range(square):
        # specify subplot and turn of axis
        ax = plt.subplot(square, square, ix)
        ax.set_xticks([])
        ax.set_yticks([])

        # plot filter channel in grayscale
        plt.imshow(feature_maps[0, :, :, ix-1], cmap='gray')
        ix += 1

        # show the figure
        plt.show()
```

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in

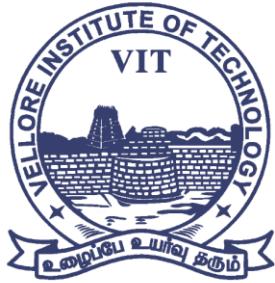
Hi, Vitta Vishnu Datta!
Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

Connected to Python 3 Google Compute Engine backend

79% 21:41 28-02-2024 ENG IN



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

[x] .. sample_data bird.jpg

✓ [32]

✓ [33] # plot feature map of first conv layer for given image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims

Connected to Python 3 Google Compute Engine backend

Disk 80.88 GB available

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta! Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

79% Search File Explorer Task View Start Taskbar ENG IN 21:42 28-02-2024

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

[x] .. sample_data bird.jpg

✓ [34] # plot feature map of first conv layer for given image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims
load the model
model = VGG16()
redefine model to output right after the first hidden layer
model = Model(inputs=model.inputs, outputs=model.layers[1].output)
model.summary()
load the image with the required shape
img = load_img('bird.jpg', target_size=(224, 224))
convert the image to an array
img = img_to_array(img)
expand dimensions so that it represents a single 'sample'
img = expand_dims(img, axis=0)
prepare the image (e.g. scale pixel values for the vgg)
img = preprocess_input(img)
get feature map for first hidden layer
feature_maps = model.predict(img)
plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
 for _ in range(square):
 # specify subplot and turn of axis
 ax = plt.subplot(square, square, ix)
 ax.set_xticks([])
 ax.set_yticks([])

✓ Connected to Python 3 Google Compute Engine backend

Disk 80.88 GB available

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta! Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

79% Search File Explorer Task View Start Taskbar ENG IN 21:42 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

```
# prepare the image (e.g. scale pixel values for the vgg)
# get feature map for first hidden layer
feature_maps = model.predict(img)
# plot all 64 maps in an 8x8 squares
square = 8
ix = 1
for _ in range(square):
    for _ in range(square):
        # specify subplot and turn off axis
        ax = plt.subplot(square, square, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        plt.imshow(feature_maps[0, :, :, ix-1], cmap='gray')
        ix += 1
    # show the figure
    plt.show()
```

Model: 'model_1'

Layer (type)	Output Shape	Param #
input_7 (InputLayer)	[None, 224, 224, 3]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792

Total params: 1792 (7.00 KB)
Trainable params: 1792 (7.00 KB)
Non-trainable params: 0 (0.00 Byte)

1/1 [=====] - 0s 62ms/step ✓ Connected to Python 3 Google Compute Engine backend

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta! Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

Disk 80.88 GB available

79% Search ENG IN 21:42 28-02-2024

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

```
# redefine model to output right after the first hidden layer
from keras.models import Model
from keras.applications.vgg16 import VGG16
```

1/1 [=====] - 0s 62ms/step ✓ Connected to Python 3 Google Compute Engine backend

vittavishnu.data2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta! Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

Disk 80.88 GB available

79% Search ENG IN 21:42 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
# redefine model to output right after the first hidden layer
from keras.models import Model
from keras.applications.vgg16 import VGG16

# Load the VGG16 model
model = VGG16()

# Get the maximum valid index
max_index = len(model.layers) - 2

# Generate even-indexed layer indices (assuming convolutional layers)
ixs = [i for i in range(1, max_index + 1) if i % 2 != 0]

# Define outputs based on valid indices
outputs = [model.layers[i].output for i in ixs]

# Create the new model
new_model = Model(inputs=model.inputs, outputs=outputs)

# Print model summary
new_model.summary()
```

Model: "model_2"

Layer (type)	Output Shape	Param #
input_8 (InputLayer)	[None, 224, 224, 3]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in
Hi, Vitta Vishnu Datta!
Manage your Google Account
Add account Sign out
Privacy Policy Terms of Service

Connected to Python 3 Google Compute Engine backend

Disk 80.88 GB available

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

```
# redefine model to output right after the first hidden layer
from keras.models import Model
from keras.applications.vgg16 import VGG16

# Load the VGG16 model
model = VGG16()

# Get the maximum valid index
max_index = len(model.layers) - 2

# Generate even-indexed layer indices (assuming convolutional layers)
ixs = [i for i in range(1, max_index + 1) if i % 2 != 0]

# Define outputs based on valid indices
outputs = [model.layers[i].output for i in ixs]

# Create the new model
new_model = Model(inputs=model.inputs, outputs=outputs)

# Print model summary
new_model.summary()
```

Model: "model_2"

Layer (type)	Output Shape	Param #
input_8 (InputLayer)	[None, 224, 224, 3]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in
Hi, Vitta Vishnu Datta!
Manage your Google Account
Add account Sign out
Privacy Policy Terms of Service

Connected to Python 3 Google Compute Engine backend

Disk 80.88 GB available



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

+ Code + Text

```
[35]: block3_conv2 (Conv2D) (None, 56, 56, 256) 590080
[35]: block3_conv3 (Conv2D) (None, 56, 56, 256) 590080
[35]: block3_pool (MaxPooling2D) (None, 28, 28, 256) 0
[35]: block4_conv1 (Conv2D) (None, 28, 28, 512) 1180160
[35]: block4_conv2 (Conv2D) (None, 28, 28, 512) 2359808
[35]: block4_conv3 (Conv2D) (None, 28, 28, 512) 2359808
[35]: block4_pool (MaxPooling2D) (None, 14, 14, 512) 0
[35]: block5_conv1 (Conv2D) (None, 14, 14, 512) 2359808
[35]: block5_conv2 (Conv2D) (None, 14, 14, 512) 2359808
[35]: block5_conv3 (Conv2D) (None, 14, 14, 512) 2359808
[35]: block5_pool (MaxPooling2D) (None, 7, 7, 512) 0
[35]: flatten (Flatten) (None, 25088) 0
[35]: fc1 (Dense) (None, 4096) 102764544
[35]: fc2 (Dense) (None, 4096) 16781312
=====
Total params: 134260544 (512.16 MB)
Trainable params: 134260544 (512.16 MB)
Non-trainable params: 0 (0.00 Byte)
```

Disk 80.88 GB available

Connected to Python 3 Google Compute Engine backend

vittavishnu.datta2021@vitstudent.ac.in Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

79% Search File Home Mail Browser Spotify 21:43 28-02-2024 ENG IN

21BAI1364 BCSE332P LAB ASSIGN.4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

+ Code + Text

```
[4]: import matplotlib.pyplot as plt
      # Assuming 'feature_maps' is a list containing feature maps from each block
      square = 8
      for fmap in feature_maps:
          # Create a new figure for each block's feature maps
          fig, axes = plt.subplots(square, square, figsize=(12, 12)) # Adjust figsize as needed
          ix = 1
          for i in range(square):
              for j in range(square):
                  # Prevent index out of bounds or unnecessary indexing
                  if ix > len(fmap):
                      # Handle the case where fewer than 64 maps are available
                      break
                  ax = axes[i, j]
                  ax.set_xticks([])
                  ax.set_yticks([])
                  # Use correct indexing to access individual maps
                  ax.imshow(fmap[ix - 1], cmap='gray') # Access entire channel
                  ix += 1
          # Tight layout to prevent overlapping labels
          fig.suptitle("Feature Maps from Block") # Optional title
          plt.tight_layout()
          plt.show()
```

Disk 80.88 GB available

Connected to Python 3 Google Compute Engine backend

vittavishnu.datta2021@vitstudent.ac.in Managed by vitstudent.ac.in

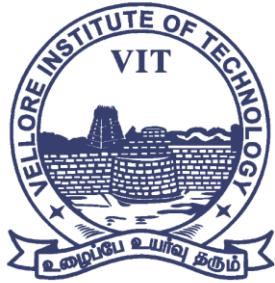
Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

79% Search File Home Mail Browser Spotify 21:43 28-02-2024 ENG IN



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

[x] sample_data bird.jpg

46

```
fig.suptitle("Feature Maps from Block") # Optional title
plt.tight_layout()
plt.show()
```

Disk 80.88 GB available

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1WEzvWBmF0vsAhuhJUJUusypS9_l0xm1NM&ec=GRBAGQM backend

ENG IN 21:44 28-02-2024

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Fe... +

File Edit View Insert Runtime Tools Help All changes saved

Files + Code + Text

[x] sample_data bird.jpg

46

```
fig.suptitle("Feature Maps from Block") # Optional title
plt.tight_layout()
plt.show()
```

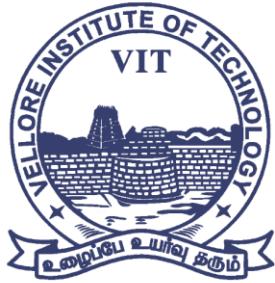
20 [38] # visualize feature maps output from each block in the vgg model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims
load the model
model = VGG16()
redefine model to output right after the first hidden layer
ixs = [2, 5, 9, 13, 17]
outputs = [model.layers[i].output for i in ixs]
model = Model(inputs=model.inputs, outputs=outputs)
load the image with the required shape
img = load_img('bird.jpg', target_size=(224, 224))
convert the image to an array
img = img_to_array(img)
expand dimensions so that it represents a single 'sample'

Connected to Python 3 Google Compute Engine backend

79% Disk 80.88 GB available

Search ...

ENG IN 21:44 28-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Feature Maps.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

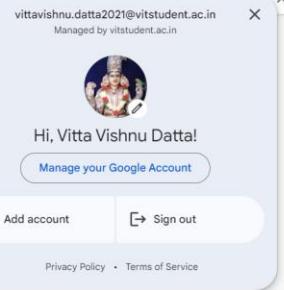
Code Text

```
# visualize feature maps output from each block in the vgg model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
from matplotlib import pyplot
from numpy import expand_dims
# load the model
model = VGG16()
# redefine model to output right after the first hidden layer
ixs = [2, 5, 9, 13, 17]
outputs = [model.layers[i].output for i in ixs]
model = Model(inputs=model.inputs, outputs=outputs)
# load the image with the required shape
img = load_img('bird.jpg', target_size=(224, 224))
# convert the image to an array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
img = expand_dims(img, axis=0)
# prepare the image (e.g. scale pixel values for the vgg)
img = preprocess_input(img)
# get feature map for first hidden layer
feature_maps = model.predict(img)
# plot the output from each block
square = 8
for fmap in feature_maps:
    # plot all 64 maps in an 8x8 squares
    ix = 1
    for _ in range(square):
        for _ in range(square):
            # show subplot and turn of axis
            ax = pyplot.subplot(square, square, ix)
            ax.set_xticks([])
            ax.set_yticks([])
            # plot filter channel in grayscale
            pyplot.imshow(fmap[0, :, :, ix-1], cmap='gray')
            ix += 1
    # show the figure
pyplot.show()
```

Connected to Python 3 Google Compute Engine backend

79% Search File Explorer Home Run Task Manager Task View Taskbar

ENG IN 21:44 28-02-2024



21BAI1364 BCSE332P LAB ASSIG... How to Visualize Filters and Feature Maps.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

[x] ..

[x] sample_data

[x] bird.jpg

Code Text

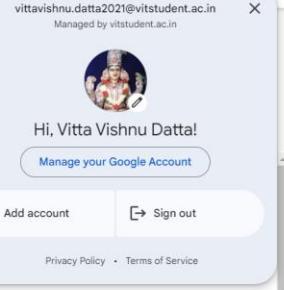
```
# specify subplot and turn of axis
ax = pyplot.subplot(square, square, ix)
ax.set_xticks([])
ax.set_yticks([])
# plot filter channel in grayscale
pyplot.imshow(fmap[0, :, :, ix-1], cmap='gray')
ix += 1
# show the figure
pyplot.show()
```

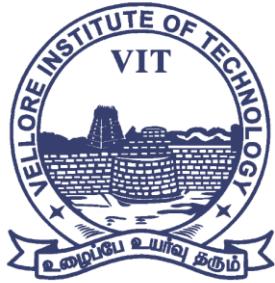
1/1 [=====] - 1s 725ms/step

Connected to Python 3 Google Compute Engine backend

79% Search File Explorer Home Run Task Manager Task View Taskbar

ENG IN 21:45 28-02-2024





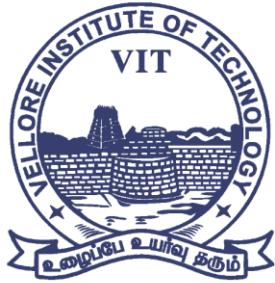
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The notebook interface shows a grid of small bird images. On the left, there's a file browser with "sample_data" and "bird.jpg" selected. The status bar indicates "Connected to Python 3 Google Compute Engine backend". On the right, a sidebar displays a Google account profile for "vittavishnu.data2021@vitstudent.ac.in". The desktop taskbar at the bottom shows various application icons.

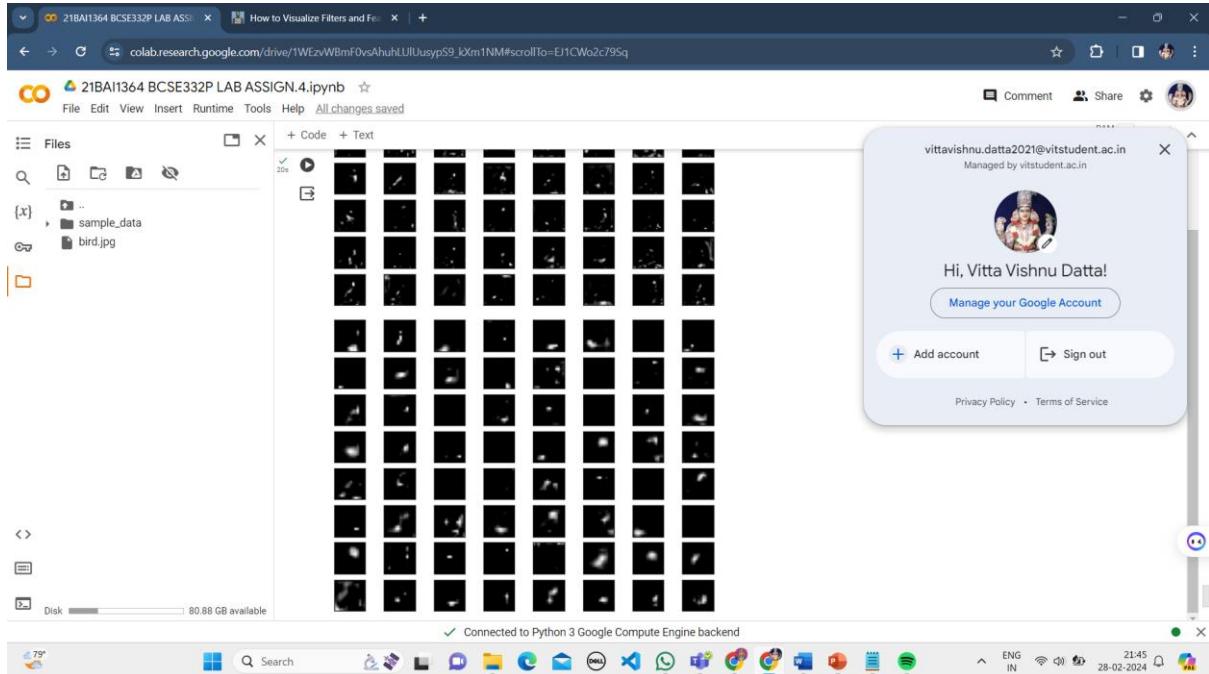
A second screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". This view is nearly identical to the first, showing the same grid of bird images, file browser, and connected backend status. The Google account sidebar and desktop taskbar are also present.



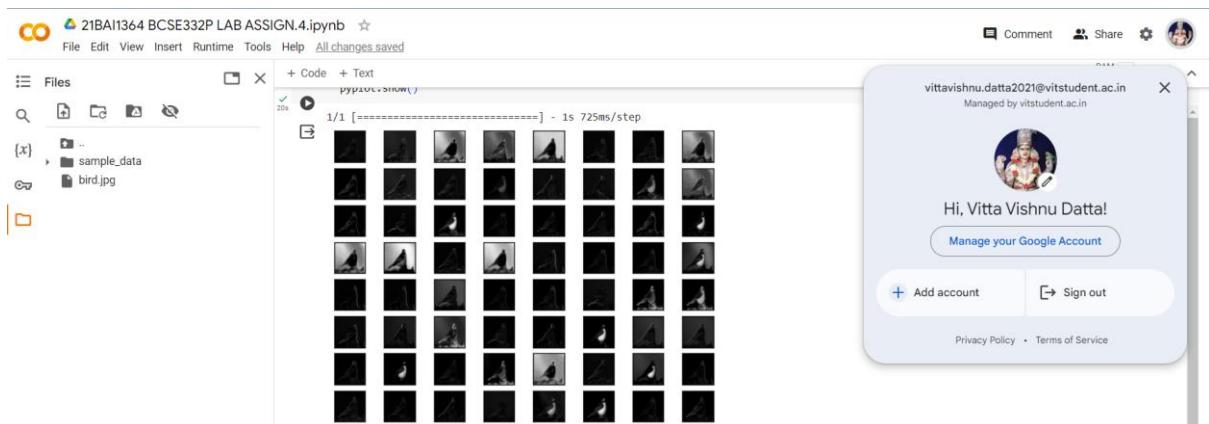
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



Visualization of the Feature Maps Extracted From Block 1 in the VGG16 Model



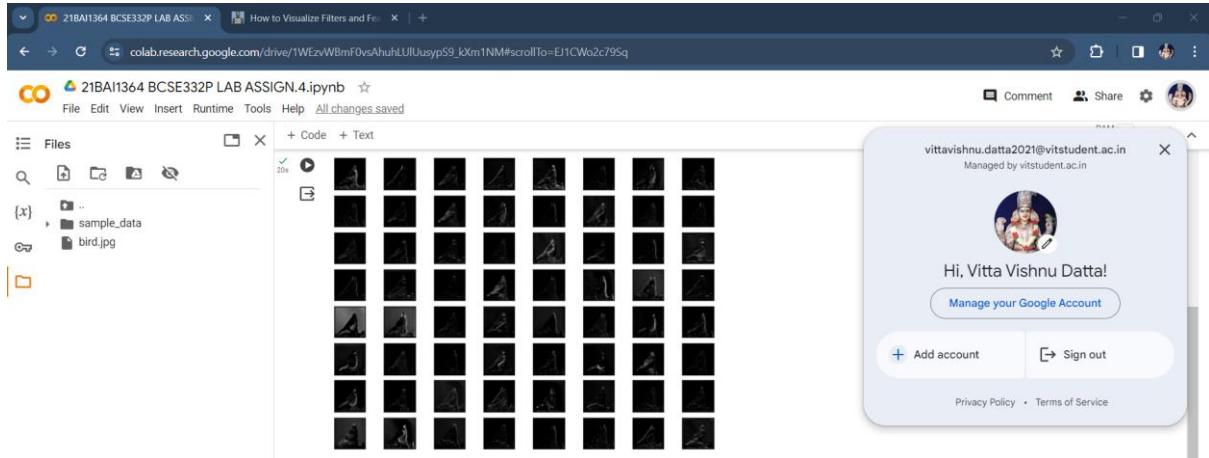
Visualization of the Feature Maps Extracted From Block 2 in the VGG16 Model



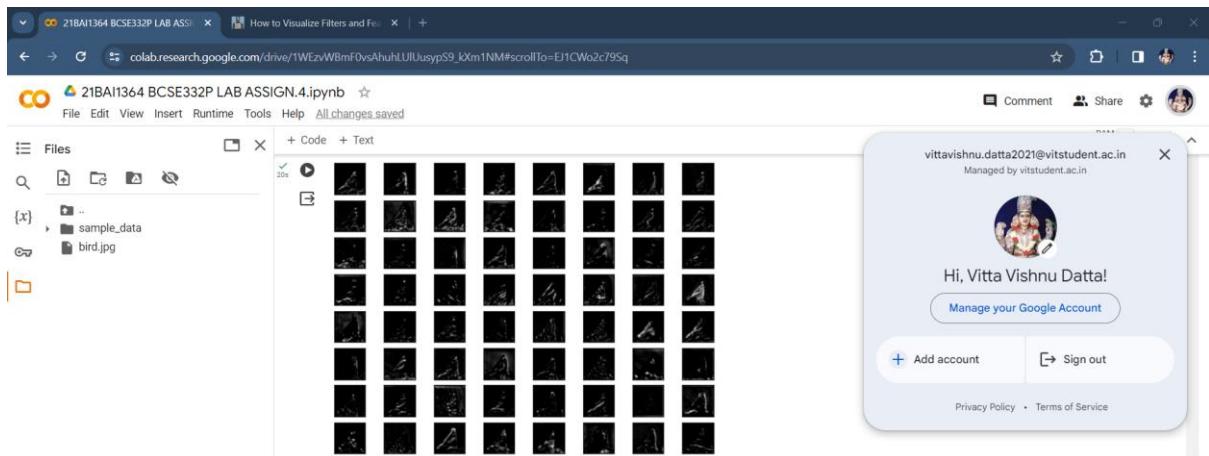
VIT®

Vellore Institute of Technology

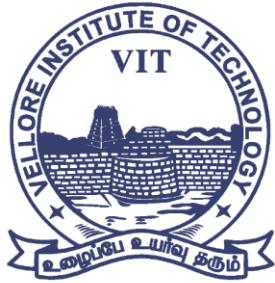
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



Visualization of the Feature Maps Extracted From Block 3 in the VGG16 Model



Visualization of the Feature Maps Extracted From Block 4 in the VGG16 Model



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Jupyter Notebook in Google Colab. The notebook title is "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell displays a grid of 38 feature maps, each showing a different internal representation of a bird image. To the right of the notebook, a Google account sidebar for "vittavishnu.datta2021@vitstudent.ac.in" is visible, showing a profile picture, the greeting "Hi, Vitta Vishnu Datta!", and links for "Manage your Google Account", "Add account", and "Sign out".

Visualization of the Feature Maps Extracted From Block 5 in the VGG16 Model

A screenshot of a Jupyter Notebook in Google Colab, identical to the one above it. It shows a grid of 38 feature maps from a VGG16 model's Block 5. The sidebar on the right shows the same Google account information for "vittavishnu.datta2021@vitstudent.ac.in".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

CODE FOR PRE-TRAINED VGG 16 MODEL:

```
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load the CIFAR-10 dataset
(train_data, val_data), info = tfds.load(
    'cifar10',
    split=['train[:80%]', 'train[80%:]'],
    with_info=True,
    as_supervised=True
)

# Resize images to match AlexNet's input size (227x227 pixels)
def preprocess_image(image, label):
    image = tf.image.resize(image, (227, 227))
    return image, label

train_data = train_data.map(preprocess_image)
val_data = val_data.map(preprocess_image)
# Define AlexNet-like architecture
def alexnet_like(input_shape=(227, 227, 3), num_classes=10):
    model = models.Sequential()

    # Adjust the layers for smaller input size
    model.add(layers.Conv2D(96, (11, 11), strides=(4, 4),
activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Conv2D(256, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Conv2D(384, (3, 3), activation='relu'))
    model.add(layers.Conv2D(384, (3, 3), activation='relu'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))

model.add(layers.Dense(num_classes, activation='softmax'))

return model

import tensorflow as tf

# Assuming `alexnet_like` function exists and defines the model
architecture

# Define placeholder variables for data and labels (customize based on
your data)
image_data = tf.keras.Input(shape=(224, 224, 3)) # Example image input
shape (adjust as needed)
label_data = tf.keras.Input(shape=(1,)) # Example label shape (adjust
as needed)

# Define your training data loading and preprocessing logic here
(replace with your specific code)
# You can use libraries like pandas, NumPy, or image processing
libraries to load and preprocess your training data.
# For demonstration purposes, we'll use placeholder data:
train_images = tf.random.normal(shape=(10000, 224, 224, 3)) # Replace
with your actual training images
train_labels = tf.random.uniform(shape=(10000,), minval=0, maxval=9,
dtype=tf.int32) # Replace with your actual labels

# Create the model using the `alexnet_like` function
model = alexnet_like(inputs=image_data)

# Compile the model with appropriate loss and optimizer
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Define batch size and number of epochs
batch_size = 128
epochs = 10
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Create TensorFlow datasets from training data, assuming proper
preprocessing is done

train_dataset = tf.data.Dataset.from_tensor_slices((train_images,
train_labels))
train_dataset =
train_dataset.shuffle(buffer_size=50000).batch(batch_size)

# Shuffle and batch validation data (assuming you have validation data)
# Replace with your validation data loading and preprocessing logic
# val_images = ... (your validation images)
# val_labels = ... (your validation labels)
# val_dataset = tf.data.Dataset.from_tensor_slices((val_images,
val_labels))
# val_dataset = val_dataset.batch(batch_size)

# Train the model
history = model.fit(train_dataset, epochs=epochs,
validation_data=None) # Replace with `val_dataset` if you have
validation data

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()

from keras.models import Sequential
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Define the AlexNet model
model = Sequential()

# Convolutional layers
model.add(Conv2D(96, (11, 11), strides=(4, 4), activation='relu',
input_shape=(224, 224, 3)))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

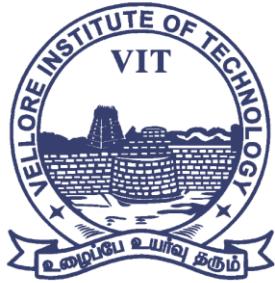
model.add(Conv2D(256, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='softmax')) # Change 1000 to the
number of classes in your problem

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Print the model summary
model.summary()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCREENSHOTS OF THE ALEX-NET MODEL:

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
[ ] import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load the CIFAR-10 dataset
(train_data, val_data), info = tfds.load(
    'cifar10',
    split=['train[80%]', 'train[20%]'],
    with_info=True,
    as_supervised=True
)

# Resize images to match AlexNet's input size (227x227 pixels)
def preprocess_image(image, label):
    image = tf.image.resize(image, (227, 227))
    return image, label

train_data = train_data.map(preprocess_image)
val_data = val_data.map(preprocess_image)
```

✓ [1] # Define AlexNet-like architecture

```
def alexnet_like(input_shape=(227, 227, 3), num_classes=10):
    model = models.Sequential()

    # Adjust the layers for smaller input size
    model.add(layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Conv2D(256, (5, 5), activation='relu'))
```

0s completed at 10:13PM

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
# Define AlexNet-like architecture
def alexnet_like(input_shape=(227, 227, 3), num_classes=10):
    model = models.Sequential()

    # Adjust the layers for smaller input size
    model.add(layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Conv2D(256, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Conv2D(384, (3, 3), activation='relu'))
    model.add(layers.Conv2D(384, (3, 3), activation='relu'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))

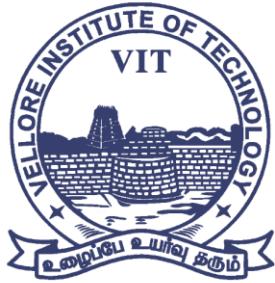
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))

    model.add(layers.Dense(num_classes, activation='softmax'))

    return model
```

import tensorflow as tf

✓ 0s completed at 10:13PM



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code in the cell is for training an AlexNet-like model. It imports TensorFlow, defines placeholder variables for data and labels, and creates training data loading logic. It then compiles the model with Adam optimizer, sparse categorical crossentropy loss, and accuracy metric. It defines batch size and epochs, creates TensorFlow datasets from training data, and trains the model. The status bar at the bottom shows "0s completed at 10:13PM".

```
import tensorflow as tf

# Assuming `alexnet_like` function exists and defines the model architecture

# Define placeholder variables for data and labels (customize based on your data)
image_data = tf.keras.Input(shape=(224, 224, 3)) # Example image input shape (adjust as needed)
label_data = tf.keras.Input(shape=(1,)) # Example label shape (adjust as needed)

# Define your training data loading and preprocessing logic here (replace with your specific code)
# You can use libraries like pandas, NumPy, or image processing libraries to load and preprocess your training data.
# For demonstration purposes, we'll use placeholder data:
train_images = tf.random.normal(shape=(10000, 224, 224, 3)) # Replace with your actual training images
train_labels = tf.random.uniform(shape=(10000,), minval=0, maxval=9, dtype=tf.int32) # Replace with your actual labels

# Create the model using the `alexnet_like` function
model = alexnet_like(inputs=image_data)

# Compile the model with appropriate loss and optimizer
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Define batch size and number of epochs
batch_size = 128
epochs = 10

# Create TensorFlow datasets from training data, assuming proper preprocessing is done
train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
train_dataset = train_dataset.shuffle(buffer_size=50000).batch(batch_size)

# Shuffle and batch validation data (assuming you have validation data)
# Replace with your validation data loading and preprocessing logic
# val_images = ... (your validation images)
# val_labels = ... (your validation labels)
# val_dataset = tf.data.Dataset.from_tensor_slices((val_images, val_labels))
# val_dataset = val_dataset.batch(batch_size)

# Train the model
history = model.fit(train_dataset, epochs=epochs, validation_data=None) # Replace with `val_dataset` if you have validation data

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code is identical to the one in the previous screenshot, but a tooltip for the user "Vitta Vishnu Datta 21BAI1364 vittavishnu.datta2021@vitstudent.ac.in" is visible in the top right corner. The status bar at the bottom shows "0s completed at 10:13PM".

```
import tensorflow as tf

# Assuming `alexnet_like` function exists and defines the model architecture

# Define placeholder variables for data and labels (customize based on your data)
image_data = tf.keras.Input(shape=(224, 224, 3)) # Example image input shape (adjust as needed)
label_data = tf.keras.Input(shape=(1,)) # Example label shape (adjust as needed)

# Define your training data loading and preprocessing logic here (replace with your specific code)
# You can use libraries like pandas, NumPy, or image processing libraries to load and preprocess your training data.
# For demonstration purposes, we'll use placeholder data:
train_images = tf.random.normal(shape=(10000, 224, 224, 3)) # Replace with your actual training images
train_labels = tf.random.uniform(shape=(10000,), minval=0, maxval=9, dtype=tf.int32) # Replace with your actual labels

# Create the model using the `alexnet_like` function
model = alexnet_like(inputs=image_data)

# Compile the model with appropriate loss and optimizer
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

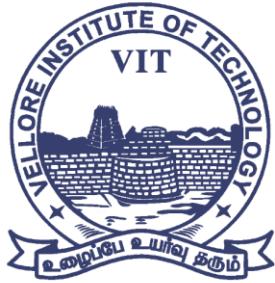
# Define batch size and number of epochs
batch_size = 128
epochs = 10

# Create TensorFlow datasets from training data, assuming proper preprocessing is done
train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
train_dataset = train_dataset.shuffle(buffer_size=50000).batch(batch_size)

# Shuffle and batch validation data (assuming you have validation data)
# Replace with your validation data loading and preprocessing logic
# val_images = ... (your validation images)
# val_labels = ... (your validation labels)
# val_dataset = tf.data.Dataset.from_tensor_slices((val_images, val_labels))
# val_dataset = val_dataset.batch(batch_size)

# Train the model
history = model.fit(train_dataset, epochs=epochs, validation_data=None) # Replace with `val_dataset` if you have validation data

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Train the model
history = model.fit(train_dataset, epochs=epochs, validation_data=None) # Replace with `val_dataset` if you have validation data

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
# Uncomment the following line if you have validation data
# plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Define the AlexNet model
model = Sequential()

# Convolutional layers
model.add(Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

model.add(Conv2D(256, (5, 5), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='softmax')) # Change 1000 to the number of classes in your problem

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Print the model summary
model.summary()
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Print the model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 54, 54, 96)	34944
max_pooling2d_3 (MaxPooling2D)	(None, 26, 26, 96)	0
conv2d_4 (Conv2D)	(None, 26, 26, 256)	614656
max_pooling2d_4 (MaxPooling2D)	(None, 12, 12, 256)	0
conv2d_5 (Conv2D)	(None, 12, 12, 384)	885120
conv2d_6 (Conv2D)	(None, 12, 12, 384)	1327488
conv2d_7 (Conv2D)	(None, 12, 12, 256)	884992
max_pooling2d_5 (MaxPooling2D)	(None, 5, 5, 256)	0
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 4096)	26218496

✓ 2s completed at 10:25PM

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

```
# max_pooling2d_3 (MaxPooling2D)
# conv2d_4 (Conv2D)
# max_pooling2d_4 (MaxPooling2D)
# conv2d_5 (Conv2D)
# conv2d_6 (Conv2D)
# conv2d_7 (Conv2D)
# max_pooling2d_5 (MaxPooling2D)
# flatten (Flatten)
# dense (Dense)
# dropout (Dropout)
# dense_1 (Dense)
# dropout_1 (Dropout)
# dense_2 (Dense)
```

Total params: 50844008 (193.95 MB)
Trainable params: 50844008 (193.95 MB)
Non-trainable params: 0 (0.00 Byte)

✓ 2s completed at 10:25PM

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
from keras.optimizers import Adam
from keras.datasets import cifar10
from keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load CIFAR-10 data for demonstration purposes
(x_train, y_train), (_, _) = cifar10.load_data()

# Preprocess the data
x_train = x_train.astype('float32') / 255.0
y_train = to_categorical(y_train, num_classes=10) # Assuming 10
classes for CIFAR-10

# Define the AlexNet model
model = Sequential()
# ... (as in the previous code)

# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
# ...
# Fully connected layers
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax')) # Change 10 to the number
of classes in your problem

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
# ...

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=128,
validation_split=0.2)

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell contains Python code for implementing an AlexNet CNN. The code imports necessary libraries (keras, numpy, etc.), loads CIFAR-10 data, defines the AlexNet model structure, and compiles it with Adam optimizer and categorical crossentropy loss. The code is annotated with comments explaining the steps. The Colab interface includes a file browser, a code editor, and a status bar indicating connection to a Google Compute Engine backend.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.optimizers import Adam
from keras.datasets import cifar10
from keras.utils import to_categorical
import matplotlib.pyplot as plt

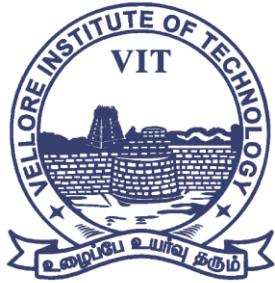
# Load CIFAR-10 data for demonstration purposes
(x_train, y_train), (_, _) = cifar10.load_data()

# Preprocess the data
x_train = x_train.astype('float32') / 255.0
y_train = to_categorical(y_train, num_classes=10) # Assuming 10 classes for CIFAR-10

# Define the AlexNet model
model = Sequential()
# ... (as in the previous code)

# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
#
# Fully connected layers
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax')) # Change 10 to the number of classes in your problem

# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# ...

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=128, validation_split=0.2)

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Connected to Python 3 Google Compute Engine backend

Disk 80.09 GB available

28-02-2024 23:38 ENG IN

```
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Connected to Python 3 Google Compute Engine backend

Disk 80.09 GB available

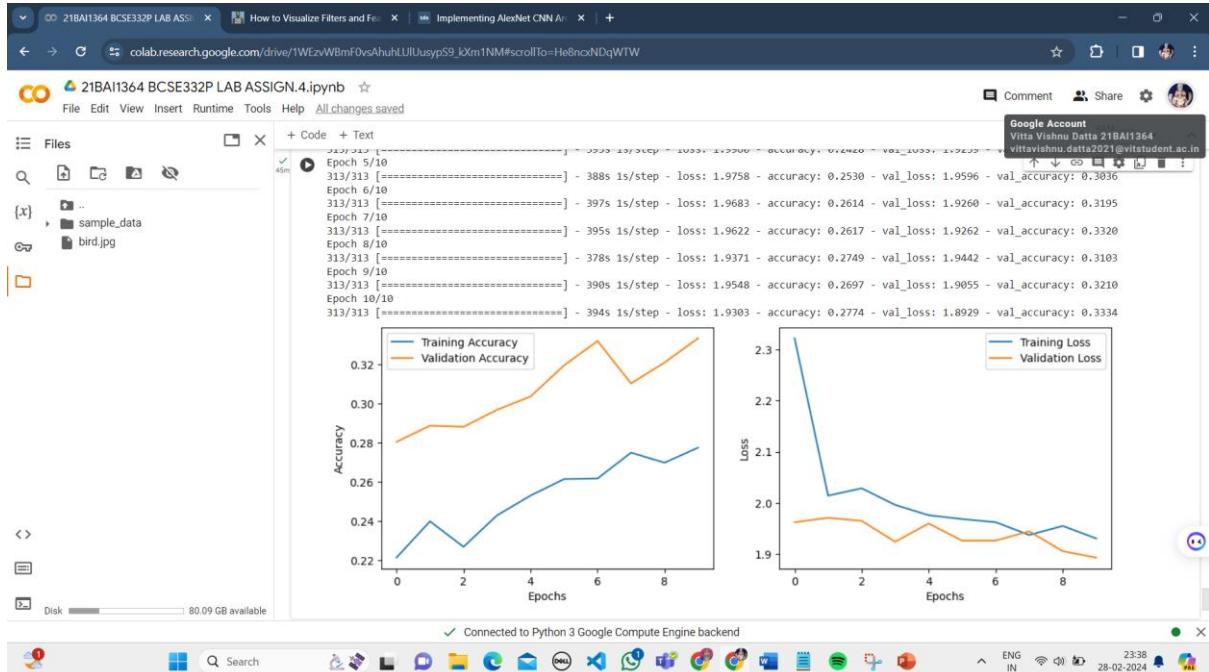
28-02-2024 23:38 ENG IN



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



4. Design a new CNN model with your dataset

CODE:

“DATASET CONSIDERED IS CIFAR10”

```
from keras.datasets import cifar10
# loading the dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# Imports for the dataset and building the neural network
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten
from tensorflow.keras.utils import to_categorical # Import
to_categorical

# Load the dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Preprocess the data
```



```
X_train = X_train.reshape(X_train.shape[0], 32, 32, 3)
X_test = X_test.reshape(X_test.shape[0], 32, 32, 3)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# One-hot encoding using the recommended approach
n_classes = 10
print("Shape before one-hot encoding: ", y_train.shape)
Y_train = to_categorical(y_train, n_classes)
Y_test = to_categorical(y_test, n_classes)
print("Shape after one-hot encoding: ", Y_train.shape)

# Build the CNN model
model = Sequential()

# convolutional layer
model.add(Conv2D(50, kernel_size=(3,3), strides=(1,1), padding='same',
activation='relu', input_shape=(32, 32, 3)))

# convolutional layer
model.add(Conv2D(75, kernel_size=(3,3), strides=(1,1), padding='same',
activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(125, kernel_size=(3,3), strides=(1,1), padding='same',
activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# flatten output of conv
model.add(Flatten())

# hidden layer
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.3))

# output layer
```



VIT®

Vellore Institute of Technology

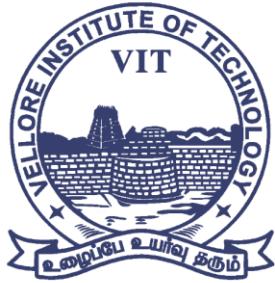
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model.add(Dense(10, activation='softmax'))\n\n# Compile and train the model\nmodel.compile(loss='categorical_crossentropy', metrics=['accuracy'],\noptimizer='adam')\nmodel.fit(X_train, Y_train, batch_size=128, epochs=10,\nvalidation_data=(X_test, Y_test))
```

SCREENSHOTS OF THE CIFAR 10 CNN MODEL:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell contains Python code for loading the CIFAR-10 dataset and building a neural network. A Google sign-in dialog is overlaid on the right side of the screen, displaying the user's profile picture and name, "Hi, Vitta Vishnu Datta!". The dialog also includes links for "Manage your Google Account", "Add account", and "Sign out". The status bar at the bottom shows the weather as "24°C Mostly sunny" and the date/time as "29-02-2024 07:58".

```
# Imports for the dataset and building the neural network\nfrom keras.datasets import cifar10\n# loading the dataset\n(X_train, y_train), (X_test, y_test) = cifar10.load_data()\n\nDownloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz\n170498071/170498071 [=====] - 2s 0us/step\n\n# Load the dataset\n(X_train, y_train), (X_test, y_test) = cifar10.load_data()\n\n# Preprocess the data\nX_train = X_train.reshape(X_train.shape[0], 32, 32, 3)\nX_test = X_test.reshape(X_test.shape[0], 32, 32, 3)\nX_train = X_train.astype('float32')\nX_test = X_test.astype('float32')\nX_train /= 255\nX_test /= 255\n\n# One-hot encoding using the recommended approach\nn_classes = 10\nprint("Shape before one-hot encoding: ", y_train.shape)\ny_train = to_categorical(y_train, n_classes)\ny_test = to_categorical(y_test, n_classes)\nprint("Shape after one-hot encoding: ", y_train.shape)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
print("Shape before one-hot encoding: ", y_train.shape)
y_train = to_categorical(y_train, n_classes)
y_test = to_categorical(y_test, n_classes)
print("Shape after one-hot encoding: ", y_train.shape)

# Build the CNN model
model = Sequential()

# convolutional layer
model.add(Conv2D(56, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))

# convolutional layer
model.add(Conv2D(75, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(125, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# flatten output of conv
model.add(Flatten())

# hidden layer
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))

# output layer
model.add(Dense(10, activation='softmax'))
```

```
print("Shape before one-hot encoding: ", y_train.shape)
y_train = to_categorical(y_train, n_classes)
y_test = to_categorical(y_test, n_classes)
print("Shape after one-hot encoding: ", y_train.shape)

# Build the CNN model
model = Sequential()

# convolutional layer
model.add(Conv2D(56, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu', input_shape=(32, 32, 3)))

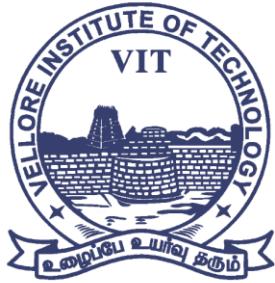
# convolutional layer
model.add(Conv2D(75, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(125, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# flatten output of conv
model.add(Flatten())

# hidden layer
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))

# output layer
model.add(Dense(10, activation='softmax'))
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code implements a neural network with two hidden layers of 500 and 256 units respectively, followed by a softmax output layer. It uses Adam optimizer and categorical crossentropy loss. The training loop shows 10 epochs of 391 steps each, with metrics like loss, accuracy, val_loss, and val_accuracy printed to the console.

```
# hidden layer
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))

# output layer
model.add(Dense(10, activation='softmax'))

# Compile and train the model
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
model.fit(X_train, Y_train, batch_size=128, epochs=10, validation_data=(X_test, Y_test))

Shape before one-hot encoding: (50000, 1)
Shape after one-hot encoding: (50000, 10)
Epoch 1/10
391/391 [=====] - 485s 1s/step - loss: 1.5496 - accuracy: 0.4290 - val_loss: 1.1231 - val_accuracy: 0.6017
Epoch 2/10
391/391 [=====] - 493s 1s/step - loss: 1.0852 - accuracy: 0.6148 - val_loss: 0.9298 - val_accuracy: 0.6760
Epoch 3/10
391/391 [=====] - 494s 1s/step - loss: 0.8981 - accuracy: 0.6875 - val_loss: 0.7952 - val_accuracy: 0.7265
Epoch 4/10
391/391 [=====] - 493s 1s/step - loss: 0.7836 - accuracy: 0.7280 - val_loss: 0.7822 - val_accuracy: 0.7286
Epoch 5/10
391/391 [=====] - 498s 1s/step - loss: 0.7002 - accuracy: 0.7552 - val_loss: 0.6957 - val_accuracy: 0.7584
Epoch 6/10
391/391 [=====] - 502s 1s/step - loss: 0.6364 - accuracy: 0.7771 - val_loss: 0.6808 - val_accuracy: 0.7591
Epoch 7/10
391/391 [=====] - 496s 1s/step - loss: 0.5717 - accuracy: 0.8002 - val_loss: 0.6727 - val_accuracy: 0.7676
Epoch 8/10
391/391 [=====] - 480s 1s/step - loss: 0.5194 - accuracy: 0.8170 - val_loss: 0.6442 - val_accuracy: 0.7788
Epoch 9/10
```

Connected to Python 3 Google Compute Engine backend

24°C High winds today ENG IN 07:59 29-02-2024

This screenshot shows the same Google Colab notebook and training process as the first one. The code and training logs are identical, demonstrating the reproducibility of the experiment.

```
# hidden layer
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))

# output layer
model.add(Dense(10, activation='softmax'))

# Compile and train the model
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
model.fit(X_train, Y_train, batch_size=128, epochs=10, validation_data=(X_test, Y_test))

Shape before one-hot encoding: (50000, 1)
Shape after one-hot encoding: (50000, 10)
Epoch 1/10
391/391 [=====] - 485s 1s/step - loss: 1.5496 - accuracy: 0.4290 - val_loss: 1.1231 - val_accuracy: 0.6017
Epoch 2/10
391/391 [=====] - 493s 1s/step - loss: 1.0852 - accuracy: 0.6148 - val_loss: 0.9298 - val_accuracy: 0.6760
Epoch 3/10
391/391 [=====] - 494s 1s/step - loss: 0.8981 - accuracy: 0.6875 - val_loss: 0.7952 - val_accuracy: 0.7265
Epoch 4/10
391/391 [=====] - 493s 1s/step - loss: 0.7836 - accuracy: 0.7280 - val_loss: 0.7822 - val_accuracy: 0.7286
Epoch 5/10
391/391 [=====] - 498s 1s/step - loss: 0.7002 - accuracy: 0.7552 - val_loss: 0.6957 - val_accuracy: 0.7584
Epoch 6/10
391/391 [=====] - 502s 1s/step - loss: 0.6364 - accuracy: 0.7771 - val_loss: 0.6808 - val_accuracy: 0.7591
Epoch 7/10
391/391 [=====] - 496s 1s/step - loss: 0.5717 - accuracy: 0.8002 - val_loss: 0.6727 - val_accuracy: 0.7676
Epoch 8/10
391/391 [=====] - 480s 1s/step - loss: 0.5194 - accuracy: 0.8170 - val_loss: 0.6442 - val_accuracy: 0.7788
Epoch 9/10
<keras/src.callbacks.History at 0x7d49d7b67d0>
```

Connected to Python 3 Google Compute Engine backend

24°C High winds today ENG IN 07:59 29-02-2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import tensorflow as tf

# Display the version
print(tf.__version__)

# other imports
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten,
Dropout
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.models import Model
# Load in the data
cifar10 = tf.keras.datasets.cifar10

# Distribute it to train and test set
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)

# Reduce pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# flatten the label values
y_train, y_test = y_train.flatten(), y_test.flatten()
# visualize data by plotting images
fig, ax = plt.subplots(5, 5)
k = 0

for i in range(5):
    for j in range(5):
        ax[i][j].imshow(x_train[k], aspect='auto')
        k += 1

plt.show()
# number of classes
K = len(set(y_train))

# calculate total number of classes
# for output layer
print("number of classes:", K)
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Build the model using the functional API
# input layer
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(i)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Flatten()(x)
x = Dropout(0.2)(x)

# Hidden layer
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)

# last hidden layer i.e.. output layer
x = Dense(K, activation='softmax')(x)

model = Model(i, x)

# model description
model.summary()
# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# Fit
r = model.fit(
    x_train, y_train, validation_data=(x_test, y_test), epochs=5)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Fit with data augmentation
# Note: if you run this AFTER calling
# the previous model.fit()
# it will CONTINUE training where it left off
batch_size = 32
data_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)

train_generator = data_generator.flow(x_train, y_train, batch_size)
steps_per_epoch = x_train.shape[0] // batch_size

r = model.fit(train_generator, validation_data=(x_test, y_test),
               steps_per_epoch=steps_per_epoch, epochs=5)

# Plot accuracy per iteration
plt.plot(r.history['accuracy'], label='acc', color='red')
plt.plot(r.history['val_accuracy'], label='val_acc', color='green')
plt.legend()
labels = '''airplane automobile bird cat deer dog frog horse ship
truck'''.split()

# select the image from our test dataset
image_number = 0

# display the image
plt.imshow(x_test[image_number])

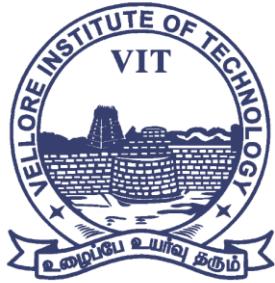
# load the image in an array
n = np.array(x_test[image_number])

# reshape it
p = n.reshape(1, 32, 32, 3)

# pass in the network for prediction and
# save the predicted label
predicted_label = labels[model.predict(p).argmax()]

# load the original label
original_label = labels[y_test[image_number]]

# display the result
print("Original label is {} and predicted label is {}".format(
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

original label, predicted label))

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
import tensorflow as tf

# Display the version
print(tf.__version__)

# other imports
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from tensorflow.keras import GlobalNormalization
from tensorflow.keras.models import Model
```

Load in the data

```
cifar10 = tf.keras.datasets.cifar10
```

Distribute it to train and test set

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)

Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>
170498871/170498871 [=====] - 4s 0us/step
(50000, 32, 32, 3) (50000, 1) (10000, 32, 32, 3) (10000, 1)

Reduce pixel values

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

flatten the label values

```
y_train, y_test = y_train.flatten(), y_test.flatten()
```

visualize data by plotting images

```
fig, ax = plt.subplots(5, 5)
k = 0

for i in range(5):
    for j in range(5):
        ax[i][j].imshow(x_train[k], aspect='auto')
        k += 1

plt.show()
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service

21BAI1364 BCSE332P LAB ASSIGN4.ipynb

```
# Reduce pixel values
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
# flatten the label values
```

```
y_train, y_test = y_train.flatten(), y_test.flatten()
```

```
# visualize data by plotting images
```

```
fig, ax = plt.subplots(5, 5)
k = 0

for i in range(5):
    for j in range(5):
        ax[i][j].imshow(x_train[k], aspect='auto')
        k += 1

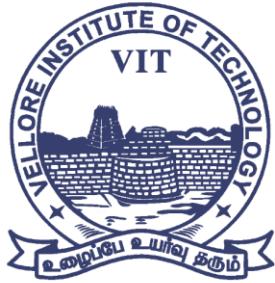
plt.show()
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



```
# number of classes
K = len(set(y_train))

# calculate total number of classes
# for output layer
print("number of classes:", K)

# Build the model using the functional API
# input layer
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(i)
x = BatchNormalization()(x)
v = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

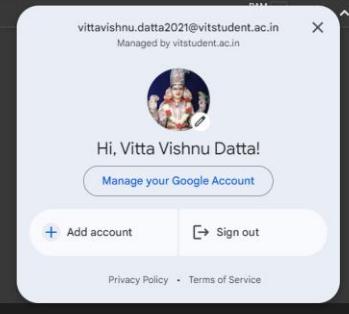
x = Flatten()(x)
x = Dropout(0.2)(x)

# Hidden layer
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)

# last hidden layer i.e.. output layer
x = Dense(K, activation='softmax')(x)

model = Model(i, x)

# model description
model.summary()
```



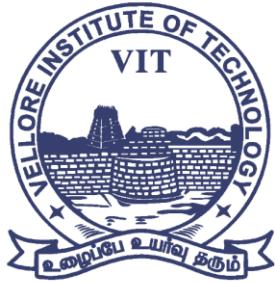
```
[5]: completed at 2:05PM
```

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
21BAI1364 BCSE332P LAB ASSIGN.4.ipynb
```

```
File Edit View Insert Runtime Tools Help All changes saved
```

```
+ Code + Text
```

```
Layer (type) Output Shape Param #
input_1 (InputLayer) [(None, 32, 32, 3)] 0
conv2d (Conv2D) (None, 32, 32, 32) 896
batch_normalization (Batch Normalization) (None, 32, 32, 32) 128
conv2d_1 (Conv2D) (None, 32, 32, 32) 9248
batch_normalization_1 (Batch Normalization) (None, 32, 32, 32) 128
max_pooling2d (MaxPooling2D) (None, 16, 16, 32) 0
conv2d_2 (Conv2D) (None, 16, 16, 64) 18496
batch_normalization_2 (Batch Normalization) (None, 16, 16, 64) 256
conv2d_3 (Conv2D) (None, 16, 16, 64) 36928
batch_normalization_3 (Batch Normalization) (None, 16, 16, 64) 256
max_pooling2d_1 (MaxPooling2D) (None, 8, 8, 64) 0
conv2d_4 (Conv2D) (None, 8, 8, 128) 73856
batch_normalization_4 (Batch Normalization) (None, 8, 8, 128) 512
```

```
Disk 81.07 GB available 1s completed at 2:05PM
```

```
vittavishnu.datta2021@vitstudent.ac.in Managed by vitstudent.ac.in
```

```
Hi, Vitta Vishnu Datta!
```

```
Manage your Google Account
```

```
+ Add account Sign out
```

```
Privacy Policy Terms of Service
```

```
Breaking news Unfolding now Search
```

```
ENG IN 29-02-2024 14:09
```

```
21BAI1364 BCSE332P LAB ASSIGN.4.ipynb
```

```
File Edit View Insert Runtime Tools Help All changes saved
```

```
+ Code + Text
```

```
[5] batch_normalization_3 (Batch Normalization) (None, 16, 16, 64) 256
max_pooling2d_1 (MaxPooling2D) (None, 8, 8, 64) 0
conv2d_4 (Conv2D) (None, 8, 8, 128) 73856
batch_normalization_4 (Batch Normalization) (None, 8, 8, 128) 512
conv2d_5 (Conv2D) (None, 8, 8, 128) 147584
batch_normalization_5 (Batch Normalization) (None, 8, 8, 128) 512
max_pooling2d_2 (MaxPooling2D) (None, 4, 4, 128) 0
flatten (Flatten) (None, 2048) 0
dropout (Dropout) (None, 2048) 0
dense (Dense) (None, 1024) 2098176
dropout_1 (Dropout) (None, 1024) 0
dense_1 (Dense) (None, 10) 10250
```

```
Total params: 2397226 (9.14 MB)
Trainable params: 2396330 (9.14 MB)
Non-trainable params: 896 (3.50 KB)
```

```
Disk 81.07 GB available 1s completed at 2:05PM
```

```
vittavishnu.datta2021@vitstudent.ac.in Managed by vitstudent.ac.in
```

```
Hi, Vitta Vishnu Datta!
```

```
Manage your Google Account
```

```
+ Add account Sign out
```

```
Privacy Policy Terms of Service
```

```
91°F Sunny Search
```

```
ENG IN 29-02-2024 14:10
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Fit
r = model.fit(
    x_train, y_train, validation_data=(x_test, y_test), epochs=5)

Epoch 1/5
1563/1563 [=====] - 449s 287ms/step - loss: 1.2046 - accuracy: 0.5728 - val_loss: 0.6567 - val_accuracy: 0.7857
Epoch 2/5
1563/1563 [=====] - 466s 298ms/step - loss: 0.8221 - accuracy: 0.6773 - val_loss: 0.6567 - val_accuracy: 0.7716
Epoch 3/5
1563/1563 [=====] - 436s 279ms/step - loss: 0.6773 - accuracy: 0.8043 - val_loss: 0.6567 - val_accuracy: 0.7857
Epoch 4/5
1563/1563 [=====] - 439s 281ms/step - loss: 0.5728 - accuracy: 0.8043 - val_loss: 0.6567 - val_accuracy: 0.7857
Epoch 5/5
1563/1563 [=====] - 443s 283ms/step - loss: 0.4845 - accuracy: 0.8327 - val_loss: 0.7126 - val_accuracy: 0.7716

# Fit with data augmentation
# Note: if you run this AFTER calling
# the previous model.fit()
# it will CONTINUE training where it left off
batch_size = 32
data_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)
```

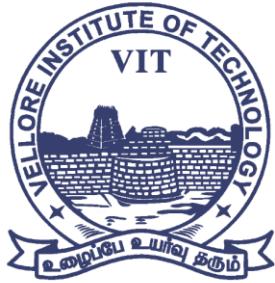
```
# Fit with data augmentation
# Note: if you run this AFTER calling
# the previous model.fit()
# it will CONTINUE training where it left off
batch_size = 32
data_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)

train_generator = data_generator.flow(x_train, y_train, batch_size)
steps_per_epoch = x_train.shape[0] // batch_size

r = model.fit(train_generator, validation_data=(x_test, y_test),
              steps_per_epoch=steps_per_epoch, epochs=5)

Epoch 1/5
1562/1562 [=====] - 467s 298ms/step - loss: 0.6774 - accuracy: 0.7699 - val_loss: 0.7753 - val_accuracy: 0.7461
Epoch 2/5
1562/1562 [=====] - 462s 296ms/step - loss: 0.6097 - accuracy: 0.7941 - val_loss: 0.6512 - val_accuracy: 0.7971
Epoch 3/5
1562/1562 [=====] - 453s 290ms/step - loss: 0.5672 - accuracy: 0.8057 - val_loss: 0.5486 - val_accuracy: 0.8159
Epoch 4/5
1562/1562 [=====] - 450s 288ms/step - loss: 0.5403 - accuracy: 0.8155 - val_loss: 0.6628 - val_accuracy: 0.7882
Epoch 5/5
1562/1562 [=====] - 449s 287ms/step - loss: 0.5054 - accuracy: 0.8286 - val_loss: 0.5864 - val_accuracy: 0.8063

[11] # Plot accuracy per iteration
plt.plot(r.history['accuracy'], label='acc', color='red')
plt.plot(r.history['val_accuracy'], label='val_acc', color='green')
plt.legend()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell [11] contains the following Python code:

```
# plot accuracy per iteration
plt.plot(r.history['accuracy'], label='acc', color='red')
plt.plot(r.history['val_accuracy'], label='val_acc', color='green')
plt.legend()
```

The resulting line plot shows two series: 'acc' (red line) and 'val_acc' (green line). The x-axis represents iterations from 0.0 to 4.0, and the y-axis represents accuracy from 0.76 to 0.82. The 'acc' series starts at approximately 0.77 and increases steadily to about 0.83. The 'val_acc' series starts at approximately 0.76, peaks at about 0.81 around iteration 2.0, dips to about 0.79 at iteration 3.0, and then rises to about 0.82 at iteration 4.0.

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.4.ipynb". The code cell [2] contains the following Python code:

```
# label mapping
labels = '''airplane automobile bird cat deer dog frog horse ship truck'''.split()

# select the image from our test dataset
image_number = 0

# display the image
plt.imshow(x_test[image_number])

# load the image in an array
n = np.array(x_test[image_number])

# reshape it
p = n.reshape(1, 32, 32, 3)

# pass in the network for prediction and
# save the predicted label
predicted_label = labels[model.predict(p).argmax()]

# load the original label
original_label = labels[y_test[image_number]]

# display the result
print("Original label is () and predicted label is ()".format(
    original_label, predicted_label))
```

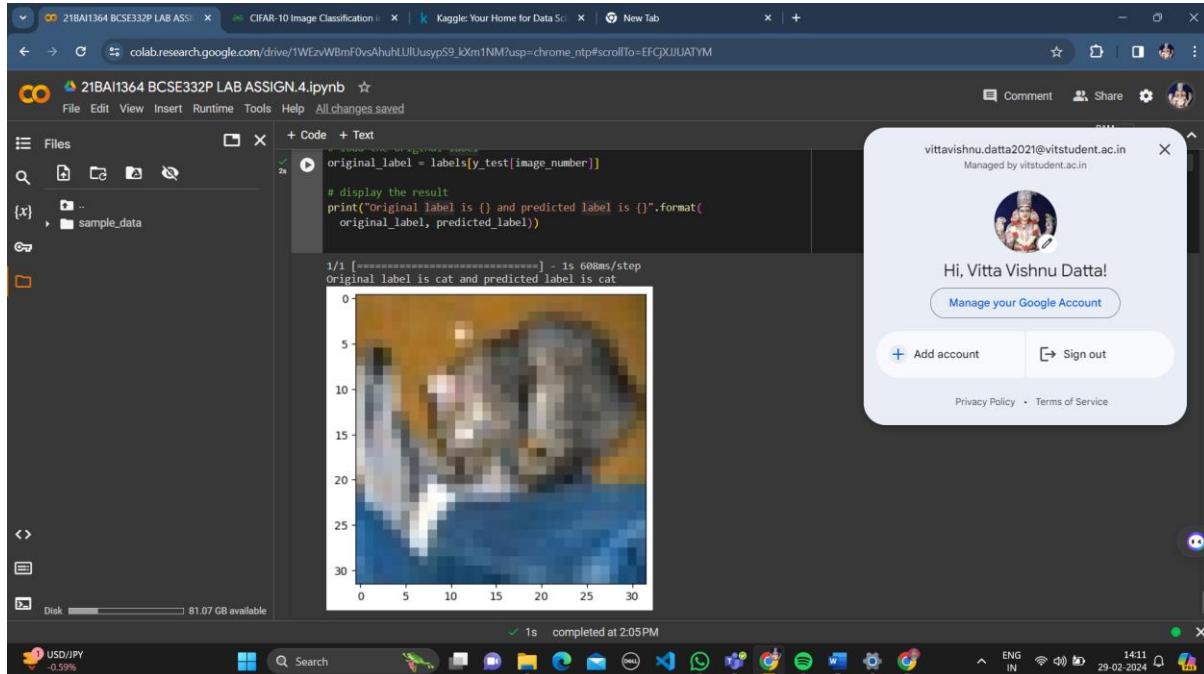
An overlay window for the Google Account "vittavishnu.datta2021@vitstudent.ac.in" is displayed. It shows a profile picture of a person in traditional Indian attire, a greeting message "Hi, Vitta Vishnu Datta!", and buttons for "Manage your Google Account", "Add account", and "Sign out". The status bar at the bottom indicates the notebook was completed at 2:05PM.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



5. Design a new fully-connected CNN model.

CODE:

```
import tensorflow as tf

# Define the model
class FullyConnectedCNN(tf.keras.Model):
    def __init__(self, input_shape, num_classes):
        super(FullyConnectedCNN, self).__init__()
        self.flatten = tf.keras.layers.Flatten()
        self.fc1 = tf.keras.layers.Dense(units=128, activation="relu")
        self.fc2 = tf.keras.layers.Dense(units=num_classes,
activation="softmax")

    def call(self, inputs):
        x = self.flatten(inputs)
        x = self.fc1(x)
        x = self.fc2(x)
        return x
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Define your data shape (replace with your actual data dimensions)
data_shape = (28, 28, 1) # Example: Assuming data is 28x28 images with
# 1 channel (grayscale)

# Create the model
model = FullyConnectedCNN(input_shape=data_shape, num_classes=10)

# Example usage (assuming you have your data prepared)
model = FullyConnectedCNN(input_shape=(data_shape[1], data_shape[2]),
                           num_classes=10) # Adjust based on your data

# Compile the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
               metrics=["accuracy"])

# Train the model (replace with your actual training data and process)
model.fit(x_train, y_train, epochs=5)

# Evaluate the model (replace with your actual test data and process)
loss, accuracy = model.evaluate(x_test, y_test)
print("Test accuracy:", accuracy)
```

SCREENSHOTS OF THE CODE SNIPPET:

```
import tensorflow as tf

# Define the model
class FullyConnectedCNN(tf.keras.Model):
    def __init__(self, input_shape, num_classes):
        super(FullyConnectedCNN, self).__init__()
        self.flatten = tf.keras.layers.Flatten()
        self.fc1 = tf.keras.layers.Dense(units=128, activation="relu")
        self.fc2 = tf.keras.layers.Dense(units=num_classes, activation="softmax")

    def call(self, inputs):
        x = self.flatten(inputs)
        x = self.fc1(x)
        x = self.fc2(x)
        return x

# Define your data shape (replace with your actual data dimensions)
data_shape = (28, 28, 1) # Example: Assuming data is 28x28 images with 1 channel (grayscale)

# Create the model
model = FullyConnectedCNN(input_shape=data_shape, num_classes=10)

# Example usage (assuming you have your data prepared)
model = FullyConnectedCNN(input_shape=(data_shape[1], data_shape[2]), num_classes=10) # Adjust based on your data

# compile the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# Train the model (replace with your actual training data and process)
model.fit(x_train, y_train, epochs=5)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Example usage (assuming you have your data prepared)
model = FullyConnectedCNN(input_shape=(data_shape[1], data_shape[2]), num_classes=10) # Adjust based on your data

# Compile the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# Train the model (replace with your actual training data and process)
model.fit(x_train, y_train, epochs=5)

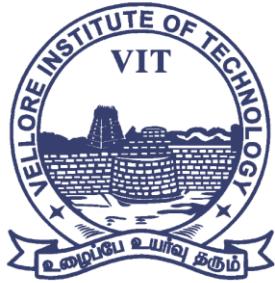
# Evaluate the model (replace with your actual test data and process)
loss, accuracy = model.evaluate(x_test, y_test)
print("Test accuracy:", accuracy)
```

Epoch 1/5
1563/1563 [=====] - 11s 6ms/step - loss: 1.8880 - accuracy: 0.3216
Epoch 2/5
1563/1563 [=====] - 11s 7ms/step - loss: 1.7506 - accuracy: 0.3728
Epoch 3/5
1563/1563 [=====] - 11s 7ms/step - loss: 1.7006 - accuracy: 0.3920
Epoch 4/5
1563/1563 [=====] - 11s 7ms/step - loss: 1.6699 - accuracy: 0.4051
Epoch 5/5
1563/1563 [=====] - 11s 7ms/step - loss: 1.6439 - accuracy: 0.4147
313/313 [=====] - 1s 3ms/step - loss: 1.6681 - accuracy: 0.4129
Test accuracy: 0.4129000081062317

COMPARISON OF OUTPUT OF TWO CODES:

<u>CODE</u>	<u>Test Loss</u>	<u>Test Accuracy</u>	<u>CNN Error</u>
Code 1	0.05	99.95%	0.95%
Code 2	1.66	41.10%	2.60%

- Code 1 refers to the CNN model built by the cifar10 dataset.
- Code 2 refers to the CNN model which is a fully-connected.



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



PREPARED AND SUBMITTED BY:

VITTA VISHNU DATTA.

21BAI1364 – SCOPE.

VIT – CHENNAI CAMPUS.