Week-10 : Construct VGG16 network, transfer the pre trained weights from Imagenet for classification of the cats and dogs.

Code :

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input,
decode_predictions
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

# Load the VGG16 model pre-trained on ImageNet data
base_model = VGG16(weights='imagenet', include_top=False)

# Freeze the layers of the pre-trained model
for layer in base_model.layers:
    layer.trainable = False

# Add custom top layers for cats and dogs classification
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(2, activation='softmax')(x)  # 2 classes: cats
and dogs

# Combine the base model with top layers
model = Model(inputs=base_model.input, outputs=predictions)

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

# Data augmentation for train set
train_datagen = ImageDataGenerator(rescale=1./255,
shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

# Data augmentation for test set (only rescaling)
```

```
test_datagen = ImageDataGenerator(rescale=1. / 255)

# Load and preprocess the training data
train_generator = train_datagen.flow_from_directory(
    r'C:\Users\Zai\Untitled Folder\training_set\training_set',  # path
to the training data directory
    target_size=(224, 224),  # resize images to fit VGG16 input size
    batch_size=32,
    class_mode='categorical')  # 2 classes: cats and dogs

    # Load and preprocess the test data
    validation_generator = test_datagen.flow_from_directory(
        r'C:\Users\Zai\Untitled Folder\test_set\test_set',  # path to the
validation data directory
        target_size=(224, 224),  # resize images to fit VGG16 input
size
        batch_size=32,
        class_mode='categorical')  # 2 classes: cats and dogs

    # Train the model
    model.fit(
        train_generator,
        steps_per_epoch=2000 // 32,  # number of training images //
batch size
        epochs=10,
        validation_data=validation_generator,
        validation_steps=800 // 32 )  # number of validation images //
batch size

    # Save the model
    model.save('cats_and_dogs_classification_model.h5')
```

Output :

```
Epoch 1/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 1929s 31s/step - accuracy: 0.6642 - loss: 0.6154
 - val_accuracy: 0.8600 - val_loss: 0.3294

Epoch 2/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 2246s 36s/step - accuracy: 0.8624 - loss: 0.3245
 - val_accuracy: 0.8587 - val_loss: 0.3173

Epoch 3/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 1830s 30s/step - accuracy: 0.8762 - loss: 0.2941
 - val_accuracy: 0.9031 - val_loss: 0.2180

Epoch 4/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 1692s 28s/step - accuracy: 0.8813 - loss: 0.2627
 - val_accuracy: 0.8788 - val_loss: 0.2542

Epoch 5/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 561s 9s/step - accuracy: 0.8745 - loss: 0.2714 -
 val_accuracy: 0.8988 - val_loss: 0.2373

Epoch 6/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 1662s 27s/step - accuracy: 0.8986 - loss: 0.2294
 - val_accuracy: 0.9149 - val_loss: 0.1937

Epoch 7/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 2132s 35s/step - accuracy: 0.9022 - loss: 0.2325
 - val_accuracy: 0.9237 - val_loss: 0.1752

Epoch 8/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 2329s 38s/step - accuracy: 0.8893 - loss: 0.2577
 - val_accuracy: 0.8712 - val_loss: 0.2990

Epoch 9/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 2017s 32s/step - accuracy: 0.9202 - loss: 0.2106
 - val_accuracy: 0.8913 - val_loss: 0.2459

Epoch 10/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 639s 10s/step - accuracy: 0.8826 - loss: 0.2475
- val_accuracy: 0.9187 - val_loss: 0.2001
```