Source Code

modelTrainer.py :

```
# import necessary layers
from tensorflow.keras import layers, models
import tensorflow as tf

# specify the classes to predict, the index order for each label in the model
classes = ['cat', 'dog', 'cow', 'sheep', 'horse']

# covert train image to fit for training
trdata = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1. / 255)
traindata = trdata.flow_from_directory(directory="dataset/train",
target_size=(224,224), batch_size=32, class_mode='categorical',
classes=classes)
# directory = path of train data

# 1st Conv Block
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), padding='same', activation='relu',
input_shape=(224, 224, 3)))
model.add(layers.Conv2D(16, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPool2D(pool_size=2, strides=2, padding='same'))

# 2nd Conv Block
model.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPool2D(pool_size=2, strides=2, padding='same'))

# 3nd Conv Block
model.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPool2D(pool_size=2, strides=2, padding='same'))

# Fully connected layers
model.add(layers.Flatten())

model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))

# creating the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
loss=tf.keras.losses.categorical_crossentropy, metrics=['accuracy'])
model.summary()

# train the model
model.fit(traindata, epochs=10)

# save the model
model.save("model.h5")
```

Above source code is in python language. This program of code will train a VGG8 model for image classification tasks. Model train by below code only aims to predict 5 image classes, such as cat, dog, cow, sheep and horse. Different versions of models used to compare in the report, such as VGG7, VGG6, VGG5 and VGG4, are also built based on this code through a little modification.

predictImage.py:

```
import os
import numpy as np
from keras.src.utils import load_img, img_to_array
from tensorflow.keras import models

# Load the model
loaded_model = models.load_model("model.h5")
threshold = 0.7

# Output for recording file name associated to their predict classes
output = {}
for i in range(5):
   output[i] = open(f'class_{i}.txt', 'w', encoding="utf-8")

for filename in os.listdir('dataset/test'):
   # Load the image
   image_path = os.path.join('dataset/test', filename)
   image = load_img(image_path, target_size=(224, 224))  # Assuming images are
resized to 224x224
   image_array = img_to_array(image)
   image_array = np.expand_dims(image_array, axis=0)  # Add batch dimension

   # Perform prediction
   predictions = loaded_model.predict(image_array)
   predicted_probabilities = predictions[0]

   # Filter predictions with confidence below the threshold
   if max(predicted_probabilities) >= threshold:
       # Locate the highest probabilities class
       predicted_class_indices = np.where(predicted_probabilities >=
threshold)[0]

       # Record the prediction filename for validation
       output[int(predicted_class_indices[0])].write(f'{filename}\n')

for i in range(5):
   output[i].close()
```

Above code will load the model train from modelTrainer.py, and perform prediction to the test data. Algorithm:

1) iterate each image in test data
2) preprocessing image for model process
3) predict image class
4) Threshold filtering the image irrelevant to target classes
5) record the image to its predict class
6) output a file for each class prediction

Five text files will be output, each representing a different class. Each contains the name of the image, which is predicted to be in that class.

evaluatePredict.py:

```python
# groundFile contains all file name belong to specific class
groundFile = open('val/cat.txt', 'r', encoding='utf-8')

# predictFile read the predict file which contains all file name belong to a specific class
predictFile = open('class_0.txt', 'r', encoding='utf-8')


fie = []
correct = 0
for line in groundFile:
    words = line.split()
    fie.append(words[0])

# Evaluate number of correct prediction
for line in predictFile:
    words = line.split()
    words[0] = words[0][:-4]
    if words[0] in fie:
        correct += 1

print(correct)
groundFile.close()
predictFile.close()
```

Above evaluation program requires two text file inputs.

1) Ground truth file, which contains all the file names belonging to a specific class.
2) Prediction file, which is the output from predictImage.py program.

Algorithm:

1. Read the two relative inputs, ground truth file and prediction file.
2. Compare how many correct image file names from prediction files within the ground truth file.