Deep Learning (CS 470, CS 570)

Module 4, Lecture 3: CNN Implementation

CNN Object Classification

Mount Google derive and import Python packages:

```
[1] from google.colab import drive
    # Mounting my Google drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
#Include libraries
import os
import numpy as np
import cv2
from matplotlib import pyplot as plt
from copy import deepcopy
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

#Setting the local folder path
os.getcwd()
os.chdir(r"/content/drive/My Drive/Assignment")
```

CNN Object Classification

Load datasets, normalize data, and display shape of the datasets:

```
[3] (train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

    # Normalize pixel values to be between 0 and 1
    train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 11s 0us/step
```

```
print("Shape of the training dataset, number of images and resolution:", train_images.shape)
print("Shape of the training dataset, number of images and resolution:", test_images.shape)
print("All distinct training labels:", np.unique(train_labels))
```

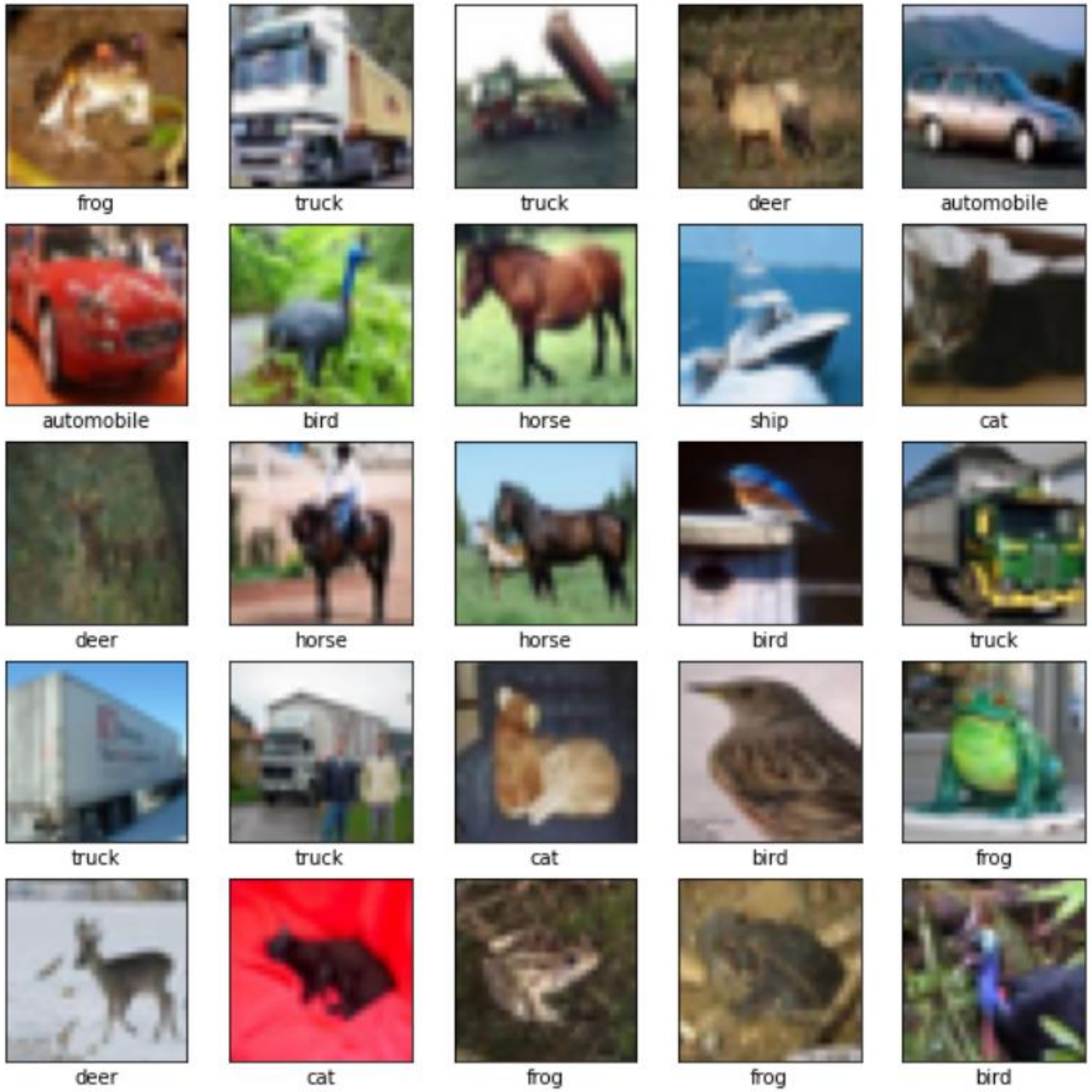
```
Shape of the training dataset, number of images and resolution: (50000, 32, 32, 3)
Shape of the training dataset, number of images and resolution: (10000, 32, 32, 3)
All distinct training labels: [0 1 2 3 4 5 6 7 8 9]
```

CNN Object Classification

Plot sample data:

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```



CNN Object Classification

Generate CNN architecture:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

2D convolution with 3D convolutional filters. Filters' depth is determined based on the depth of input image/feature map.

Reshaping the feature maps for fully connected layer.

CNN Object Classification

Output of model.summary():

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

CNN Object Classification

Training and validation of CNN model:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

CNN_trained = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.4840 - accuracy: 0.4595 - val_loss: 1.3019 - val_accuracy: 0.5403
Epoch 2/10
1563/1563 [==============================] - 65s 42ms/step - loss: 1.1186 - accuracy: 0.6060 - val_loss: 1.1216 - val_accuracy: 0.6133
Epoch 3/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.9642 - accuracy: 0.6605 - val_loss: 0.9913 - val_accuracy: 0.6519
Epoch 4/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.8678 - accuracy: 0.6956 - val_loss: 0.8788 - val_accuracy: 0.6968
Epoch 5/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.8017 - accuracy: 0.7176 - val_loss: 0.8592 - val_accuracy: 0.7061
Epoch 6/10
1563/1563 [==============================] - 64s 41ms/step - loss: 0.7474 - accuracy: 0.7369 - val_loss: 0.8448 - val_accuracy: 0.7134
Epoch 7/10
1563/1563 [==============================] - 65s 41ms/step - loss: 0.7014 - accuracy: 0.7535 - val_loss: 0.8323 - val_accuracy: 0.7149
Epoch 8/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6579 - accuracy: 0.7696 - val_loss: 0.8381 - val_accuracy: 0.7129
Epoch 9/10
1563/1563 [==============================] - 67s 43ms/step - loss: 0.6203 - accuracy: 0.7813 - val_loss: 0.8887 - val_accuracy: 0.7039
Epoch 10/10
1563/1563 [==============================] - 66s 43ms/step - loss: 0.5825 - accuracy: 0.7951 - val_loss: 0.8691 - val_accuracy: 0.7172
```