Introduction to Computer Vision HW07

- Convolution Neural Network -

2021-1학기 컴퓨터비전개론 059분반 201724437 김재현

1. Load Datasets & Normalization

CIFAR-10의 학습용 이미지와 데스트 데이터를 불러오고, 이를 정규화하여 CNN에 적용할 수 있게끔 준비하는 과정입니다. numpy.random.choice() 함수를 이용하여 50000개의 데이터 중 100개의 무작위 개별 index를 얻고, 이 index에 해당하는 data들만 subplot에 추가하여 표시하였습니다.

이후에는 다른 함수에서 쓸 수 있도록 X_train, y_train, X_test, y_test 총 4개의 값을 반환하는데, X_train 데이터만 normalize했습니다. X_test 데이터를 그대로 반환한 이유는, predict 단계 이후에 이미지를 표시할 때 (normalize되지 않은) 원본 이미지가 필요하다고 판단했기 때문입니다.

해당 코드와 sample training sets의 이미지는 아래와 같습니다.

hw07_201724437.py - Line 9

```
labels = ['airplane', 'automoblie', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
          'ship', 'truck']
```

hw07_201724437.py - Line 11~38

```
def load_and_normalize():
  fig, ax = plt.subplots(10,10, figsize=(15,15))
  ax =  ax.flatten()
  fig.tight_layout(pad=1.0)

  (X_train, y_train), (X_test, y_test) = cifar10.load_data()

  rand_index = np.random.choice(range(50000), 100)
  rand_imgs = X_train[rand_index]
  rand_label_index = y_train[rand_index].flatten()

  for i, (img, idx) in enumerate(zip(rand_imgs, rand_label_index)):
    ax[i].set_title(labels[idx], size=16)
    ax[i].imshow(img)
    ax[i].axis('off')
  plt.show()

  X_train = X_train/255.0

  return (X_train, y_train), (X_test, y_test)
```

>>

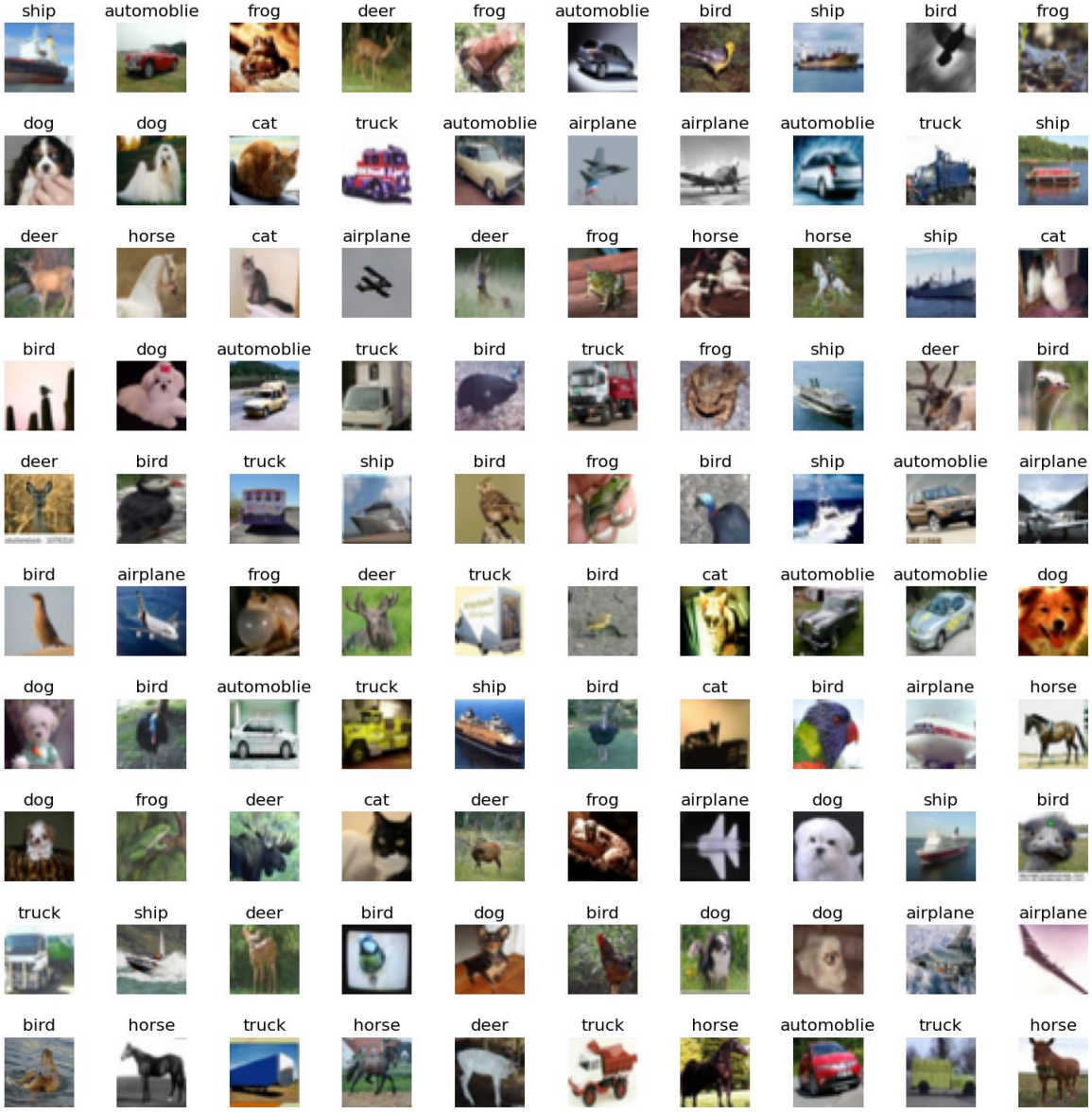

그림 1 – training_sets_sample.png

2. Training Model

keras 라이브러리를 이용하여 CNN 모델을 구성하고, 1단계에서 normalize한 데이터를 여기에 적용하여 model을 훈련시키는 과정입니다. model의 구성은 과제 파일에서 제시된 것과 동일하게 하였고, 이후 model fitting 과정에서도 epochs를 비롯한 다른 변수를 과제에 제시된 값과 같은 값을 적용하였습니다.

이 중 validation_split은 전달받은 training sets에서 얼마나 validation sets로 전환하여 사용할 것인지를 결정합니다. 여기서는 training sets 50000개 중 20%에 해당하는 10000개의 데이터를 validation set으로 전환하여, model의 적합성을 판단하는데 사용할 것입니다.

그리고 fitting이 끝난 model과 fitting 과정 중의 기록(loss, accuracy 등)은 다음 함수에서 사용할 수 있도록 반환하였습니다.

해당 코드와 model compile 결과를 나타낸 log 이미지는 아래와 같습니다. 해당 log는 Visual Studio Code 내의 내장 터미널에서 캡처하였습니다.

hw07_201724437.py - Line 40~68

```
def train_model(X_train, y_train):
  y_train_encoded = to_categorical(y_train, len(labels))

  model = Sequential(
    [
      Conv2D(32, kernel_size=5, activation='relu', input_shape=(32,32,3)),
      Conv2D(64, 5, activation='relu'),
      MaxPool2D(),
      Conv2D(128, 5, activation='relu'),
      MaxPool2D(),
      Dropout(0.25),
      Flatten(),
      Dense(128, activation='relu'),
      Dropout(0.3),
      Dense(len(labels), activation='softmax')
    ]
  )

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

  train_history = model.fit(x=X_train, y=y_train_encoded, batch_size=64,
                                              epochs=10, validation_split=0.2)

  return model, train_history
```

>>

```
터미널  문제  출력  디버그 콘솔

: NVIDIA GeForce GTX 1660 SUPER, pci bus id: 0000:07:00.0, compute capability: 7.5)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        2432
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 64)        51264
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         204928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 522,186
Trainable params: 522,186
Non-trainable params: 0
_________________________________________________________________
```

그림 2 – model_compile_summary.png

3. Evaluate & Predict

마지막으로 전달받은 fitting history를 표시하고, fitting된 모델에 새로운 test 데이터를 적용하여 model이 얼마나 잘 학습되었는지를 확인하는 단계입니다.

그래프를 표시할 때는 기존의 matplotlib.pyplot의 함수를 사용했습니다. 표시된 그래프(그림3)을 보면 epoch가 경과함에 따라 accuracy는 증가하고, loss는 감소하는 추세를 보여 model이 어느 정도 잘 학습되었다고 볼 수 있습니다.

한편 test 데이터를 적용하기 위해 1단계에서 training sets을 normalize한 것과 같이 test sets을 255로 나누어 normalize해 주었습니다. 그 다음으로 predict_classes() 함수를 이용하여 X_test 데이터에 대한 prediction, 즉 예상 label index(0 〈= i 〈= 9) 배열을 얻어내고, 이 중 12개를 임의로 추출하여 1단계와 같이 subplot에 표시해 주었습니다. 확인 결과 12개 중 10개의 데이터에 대해 prediction이 맞았음을 확인할 수 있었습니다.

해당 코드와 model fitting history 그리고 prediction sample 이미지는 아래와 같습니다.

hw07_201724437.py - Line 70~111

```
def evaluate_and_predict(model, hist, X_test, y_test):
  eval_fig, (acc_ax, loss_ax) = plt.subplots(1, 2, figsize=(8,4))

  acc_ax.plot(hist.history['accuracy'], 'blue', label='accuracy')
  acc_ax.plot(hist.history['val_accuracy'], 'orange', label='val_accuracy')
  acc_ax.legend(loc='upper left')

  loss_ax.plot(hist.history['loss'], 'blue', label='loss')
  loss_ax.plot(hist.history['val_loss'], 'orange', label='val_loss')
  loss_ax.legend(loc='upper right')

  plt.show();

  nX_test = X_test/255.0
  y_predict = model.predict_classes(nX_test, batch_size=64)

  predict_fig, predict_ax = plt.subplots(3, 4, figsize=(8,6))
  predict_ax = predict_ax.flatten()
  predict_fig.tight_layout(pad=1.0)

  rand_index = np.random.choice(range(10000), 12)
  rand_test_imgs = X_test[rand_index]
  rand_test_label_index = y_test[rand_index].flatten()
  rand_predict_label_index = y_predict[rand_index].flatten()

  for i, (img, test_idx, pred_idx) in enumerate(zip(rand_test_imgs,
                              rand_test_label_index,rand_predict_label_index)):
    predict_ax[i].axis('off')
    predict_ax[i].imshow(img)
    predict_ax[i].text(0,1,f'Prediction: {labels[pred_idx]}', backgroundcolor=
                          'white', size=6, transform=predict_ax[i].transAxes)
    predict_ax[i].text(0,0.9,f'LABEL: {labels[test_idx]}', backgroundcolor=
                              'white', size=6,transform=predict_ax[i].transAxes)
  plt.show();
```

>>

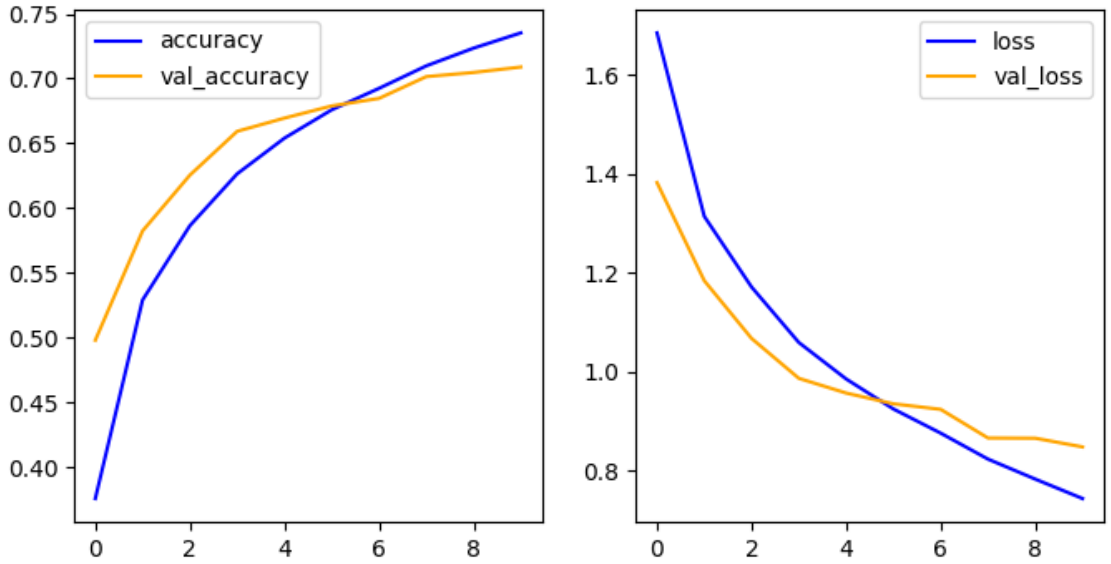

그림 3 – model_fitting_history.png

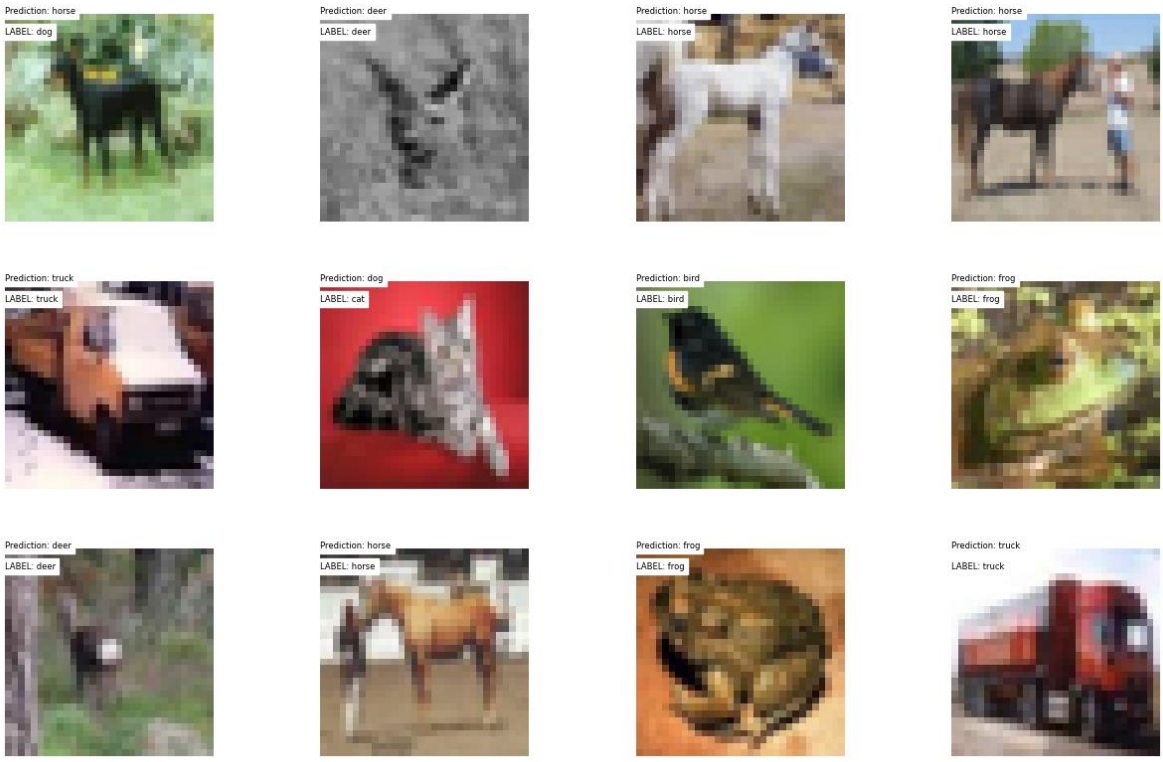

그림 4 – prediction_results_sample.png