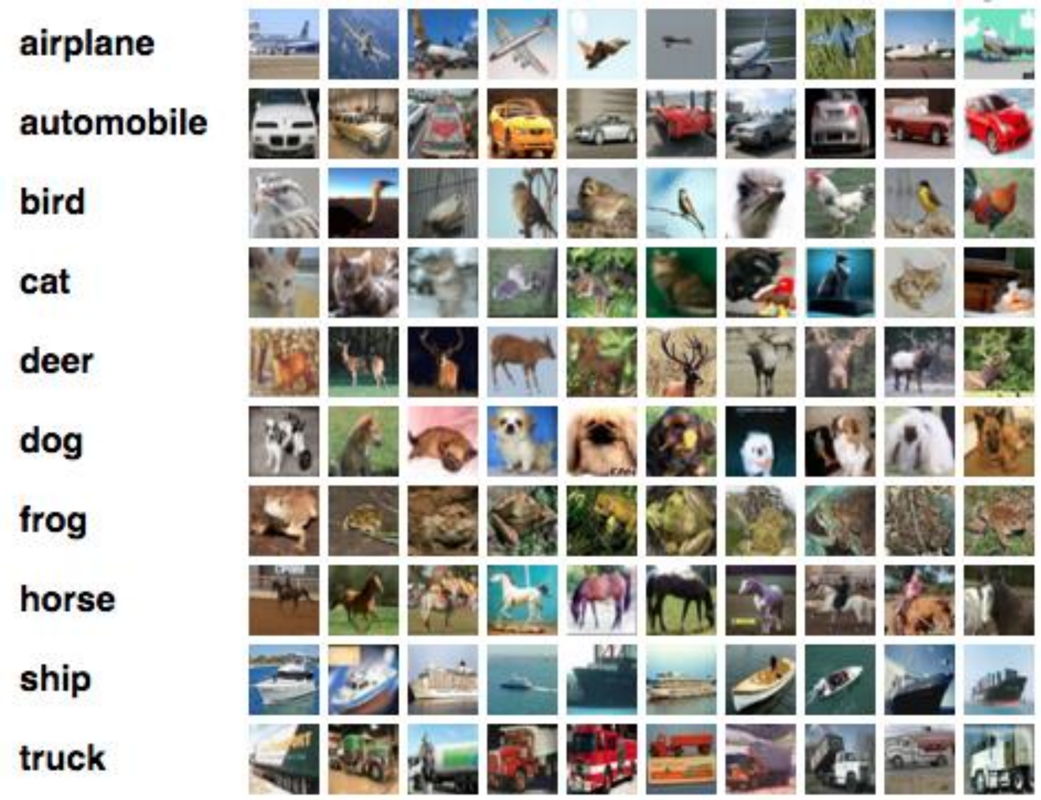
CIFAR-10 분류 구현

CIFAR-10 데이터셋

- 비행기, 자동차 등 사물의 10 개 분류
  - 손글씨와 구조하나 칼라
    - 50000개(학습용), 10000개(테스트용), 28 X 28 X 3 이미지 구조, 10개의 분류
      - 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배(ship), 트럭



파일

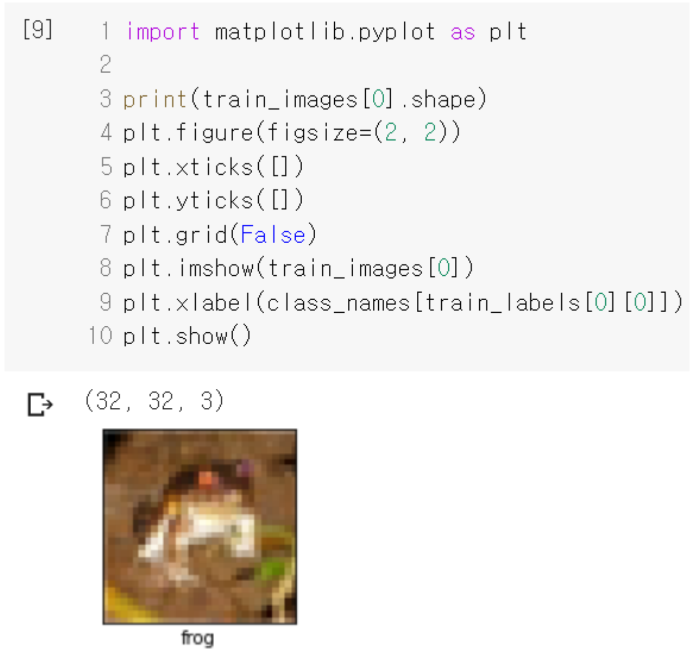
- **cifar10_basic_dnn.ipynb**

CIFAR-10 데이터 저장

- **datasets.cifar10.load_data()**

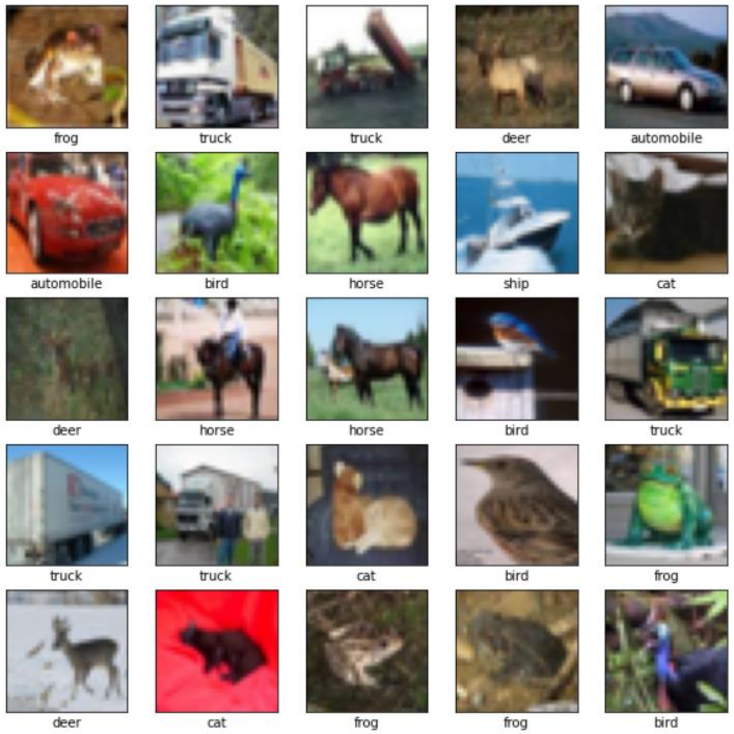
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models

cifar10 = datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

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

print("Train samples:", train_images.shape, train_labels.shape)
print("Test samples:", test_images.shape, test_labels.shape)
```

이미지 보기



```
[9]
import matplotlib.pyplot as plt

print(train_images[0].shape)
plt.figure(figsize=(2, 2))
plt.xticks([])
plt.yticks([])
plt.grid(False)
plt.imshow(train_images[0])
plt.xlabel(class_names[train_labels[0][0]])
plt.show()
```

```
(32, 32, 3)
```



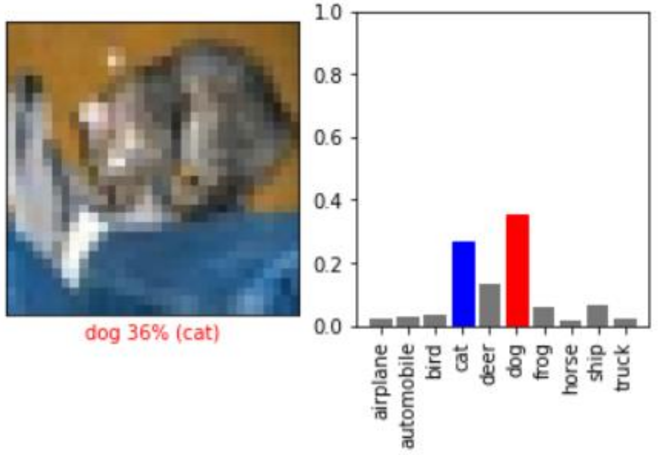
```
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()

train_images = train_images/255.0
test_images = test_images/255.0
```

모델 Sequential

```
model = models.Sequential()
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metr
ics=['accuracy'])
model.fit(train_images, train_labels, epochs=10)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
Epoch 8/10
1563/1563 [==============================] - 12s 7ms/step - loss: 1.4363 - accuracy: 0.4877
Epoch 9/10
1563/1563 [==============================] - 12s 7ms/step - loss: 1.4196 - accuracy: 0.4935
Epoch 10/10
1563/1563 [==============================] - 12s 8ms/step - loss: 1.3981 - accuracy: 0.5001
313/313 [==============================] - 1s 3ms/step - loss: 1.5084 - accuracy: 0.4705
Test accuracy: 0.47049999237060547
```

```
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img, cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                    100*np.max(predictions_array),
                                    class_names[true_label[0]]),
                                    color=color)

def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label[0]].set_color('blue')

    # 각 종류 레이블을 직접 세로로 출력
    xlabel = [class_names[i] for i in range(10)]
    plt.xticks(np.arange(10), xlabel, rotation='vertical')

predictions = model.predict(test_images)
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```

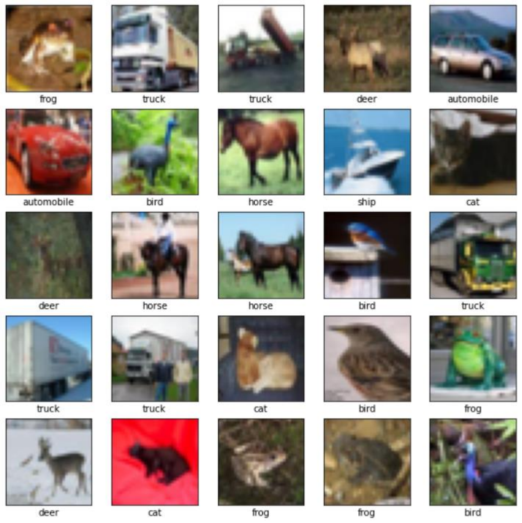


CIFAR-10 분류 CNN 구현

이미지 로드와 보기

```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models

cifar10 = datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

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

print("Train samples:", train_images.shape, train_labels.shape)
print("Test samples:", test_images.shape, test_labels.shape)

#train_images = train_images.reshape((50000, 32, 32, 3))
#test_images = test_images.reshape((10000, 32, 32, 3))


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])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

```
Train samples: (50000, 32, 32, 3) (50000, 1)
Test samples: (10000, 32, 32, 3) (10000, 1)
```



CNN 모델 생성, 학습, 평가

- **컨볼루션 신경망(convolutional neural network) 기반 이미지 분류기**

```
train_images = train_images/255.0
test_images = test_images/255.0

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, activation='softmax'))

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

model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)
```

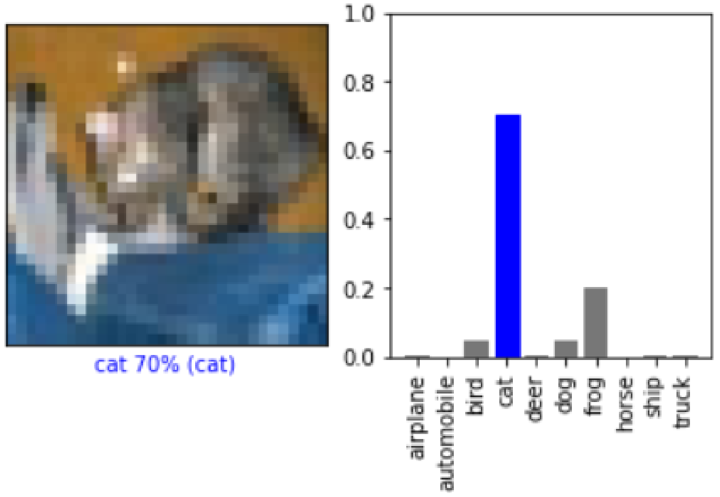
```
Epoch 10/10
1563/1563 [==============================] - 9s 6ms/step - loss: 0.5993 - accuracy: 0.7891
313/313 [==============================] - 1s 3ms/step - loss: 0.8726 - accuracy: 0.7083
Test accuracy: 0.708299994468689
```

이미지 그리기 함수

```
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img, cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label[0]]),
                                color=color)
```

확률 값 그리기

```
def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    #plt.xticks([])
    #plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label[0]].set_color('blue')

    # 각 종류 레이블을 직접 세로로 출력
    xlabel = [class_names[i] for i in range(10)]
    plt.xticks(np.arange(10), xlabel, rotation='vertical')

predictions = model.predict(test_images)
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```


cat 70% (cat)
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck