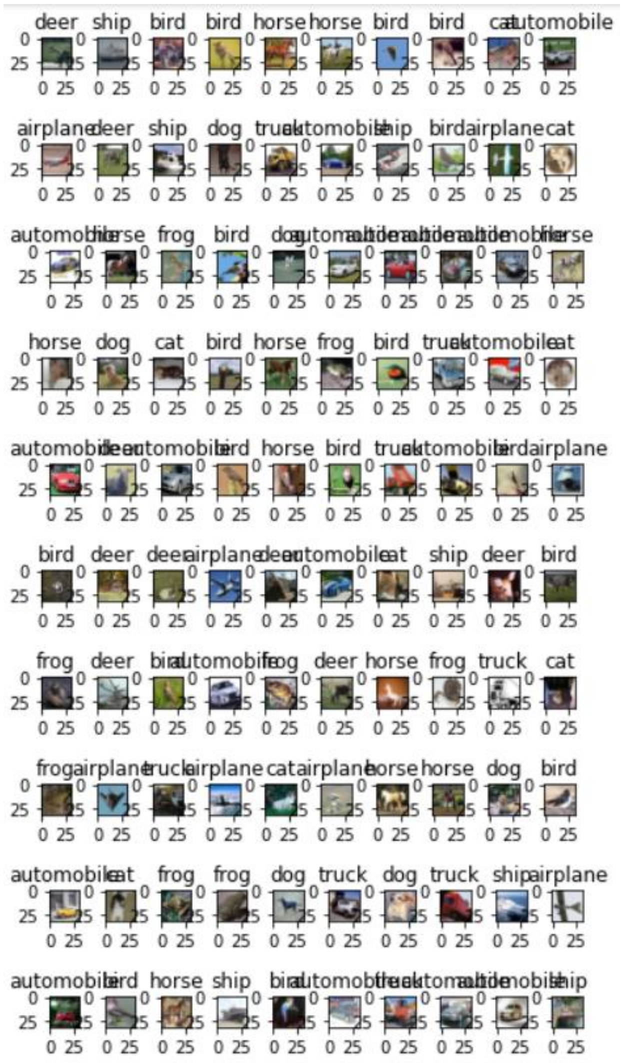
# Convolutional Neural Network

## 1. Load Datasets & Normalization

### [Code]

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import MaxPooling2D as MaxPool2D
from keras.layers.convolutional import Conv2D
from tensorflow.keras.datasets import cifar10
# 1번
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# label
labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 
'truck']
# X_train 과 그에 맞는 결과값인 X_test 100 개 쌍을 출력.
y_train # y_train[0][0]을 해야 6 이 나온다. []로 안감싸진 것이 나오게 하기 위해서는.
# 내부에 괄호 없애는 작업
fixed_y_train = []
for i in range(50000):
  fixed_y_train.append(y_train[i][0])
# 0 ~ 255 -> 0 ~ 1 로 하여 속도적인 이점을 꾀함
X_train = X_train / 255.0
X_test = X_test / 255.0
rand_list = []
for i in range(100):
  rand_list.append(np.random.randint(50000))
rand_list
# 앞에서 random 하게 train 에서 골라낸 리스트 rand_list 를 이용해서 plot 출력
index = 0
plt.figure(1)
for i in range(10):
  for j in range(10):
    plt.subplot2grid((10, 10), (i, j))
    plt.title(labels[fixed_y_train[rand_list[index]]])
    plt.imshow(X_train[rand_list[index]])
    index += 1
  plt.show()
```

### [결과]



## 2. Training Model

[Code]

```
# 2번
# sklearn의 전처리 모듈을 이용한 one-hot encoding으로 data label 변환.
from sklearn.preprocessing import OneHotEncoder
onehot_encoder = OneHotEncoder(sparse = False)
onehot_encoder.fit(y_train)
y_train = onehot_encoder.transform(y_train)
y_test = onehot_encoder.transform(y_test)
# one-hot encoding 변환 결과 확인 가능.
y_train
input_shape = (X_train.shape[1], X_train.shape[2], 1)
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),
  # class(labels) 개수
  Dense(10, activation='softmax')
])
model.compile(optimizer = 'adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
X_train.shape
y_train.shape
history = model.fit(X_train, y_train, batch_size = 64, epochs = 10, validation_split = 0.2)
```

[결과]

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 28, 28, 32)        2432
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 24, 24, 64)        51264
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 8, 8, 128)         204928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               262272
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 522,186
Trainable params: 522,186
Non-trainable params: 0
```

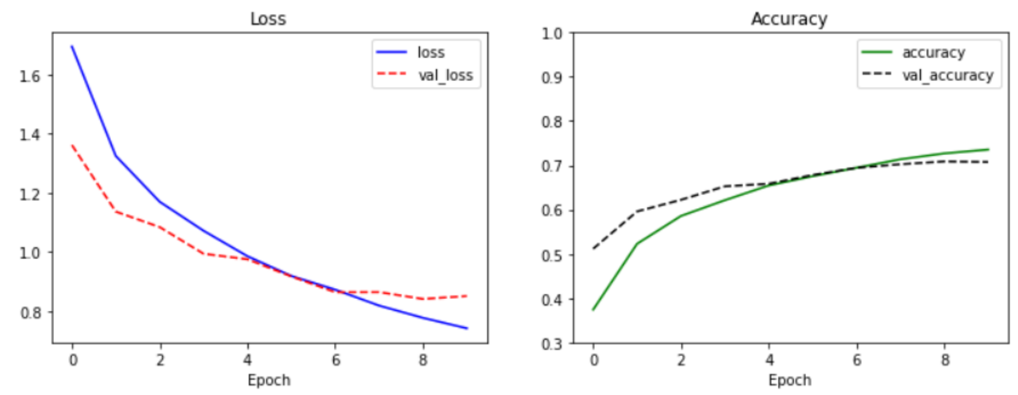
## 3. Evaluate & Predict

[Code]

```
# 3 번 평가
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.title("Loss")
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()
plt.subplot(1, 2, 2)
plt.title("Accuracy")
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'k--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.3, 1)
```

```python
plt.legend()
plt.show()
model.evaluate(X_test, y_test)
predictions = model.predict(X_test)
predictions = onehot_encoder.inverse_transform(predictions)
predictions = predictions.astype(int)
# 예측 및 실제 값 표시
fig, axes = plt.subplots(ncols = 4, nrows = 3, sharex = False, sharey = True, figsize =
(12, 9))
index = 0
for i in range(3):
    for j in range(4):
        axes[i, j].set_title('actual:' + labels[y_test[index][0]] + '\n' + 'predicted:' +
labels[predictions[index][0]])
        axes[i, j].imshow(X_test[index], cmap = 'gray')
        axes[i, j].get_xaxis().set_visible(False)
        axes[i, j].get_yaxis().set_visible(False)
        index += 1
plt.show()
```

**[결과]**



```
313/313 [==============================] - 19s 62ms/step - loss: 0.8499 - accuracy: 0.7106
[0.8499428629875183, 0.7106000185012817]
```