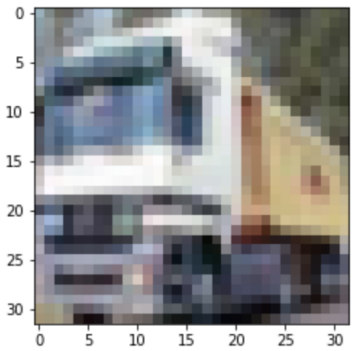
Import module and datasets.

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

(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
print(x_train.shape)

plt.imshow(x_train[1])
plt.show()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 3s 0us/step
170508288/170498071 [==============================] - 3s 0us/step
(50000, 32, 32, 3)
```



Train a regular NN model.

```
x_train, x_test = x_train / 255.0, x_test / 255.0

model = models.Sequential([
    layers.Flatten(input_shape=(32, 32, 3)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=128)
```

```
Epoch 1/5
391/391 [==============================] - 4s 8ms/step - loss: 2.1193 - accuracy: 0.2067 - val_loss: 1.9978 - val_accuracy: 0.2647
Epoch 2/5
391/391 [==============================] - 3s 7ms/step - loss: 1.9566 - accuracy: 0.2812 - val_loss: 1.9207 - val_accuracy: 0.2972
Epoch 3/5
391/391 [==============================] - 3s 7ms/step - loss: 1.9049 - accuracy: 0.3081 - val_loss: 1.8846 - val_accuracy: 0.3249
Epoch 4/5
391/391 [==============================] - 3s 7ms/step - loss: 1.8713 - accuracy: 0.3250 - val_loss: 1.8503 - val_accuracy: 0.3374
Epoch 5/5
391/391 [==============================] - 3s 7ms/step - loss: 1.8392 - accuracy: 0.3383 - val_loss: 1.8198 - val_accuracy: 0.3497
<keras.callbacks.History at 0x7f62ea84ebd0>
```

Add more convolutional layers and pooling layers, and get a summary of the model.

```
x_train, x_test = x_train / 255.0, x_test / 255.0

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.summary()
```

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_1 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_3 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten_3 (Flatten)         (None, 512)               0

 dense_9 (Dense)             (None, 64)                32832

 dense_10 (Dense)            (None, 32)                2080

 dense_11 (Dense)            (None, 10)                330

=================================================================
```

Train the above model.

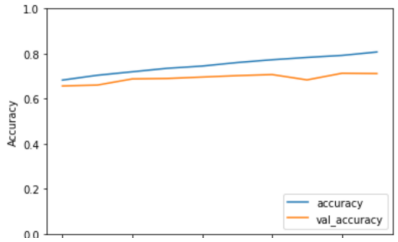
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=128)
```

```
Epoch 1/5
391/391 [==============================] - 64s 163ms/step - loss: 2.3027 - accuracy: 0.0990 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 2/5
391/391 [==============================] - 64s 163ms/step - loss: 2.3027 - accuracy: 0.0969 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 3/5
391/391 [==============================] - 64s 163ms/step - loss: 2.3027 - accuracy: 0.0971 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 4/5
391/391 [==============================] - 62s 159ms/step - loss: 2.3027 - accuracy: 0.0974 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 5/5
391/391 [==============================] - 64s 163ms/step - loss: 2.3027 - accuracy: 0.0978 - val_loss: 2.3026 - val_accuracy: 0.1000
<keras.callbacks.History at 0x7f62ea5b8a90>
```

Run it longer and visualize the accuracy.

```
391/391 [==============================] - 61s 157ms/step - loss: 0.8543 - accuracy: 0.7041 - val_loss: 0.9813 - val_accu
Epoch 3/10
391/391 [==============================] - 60s 153ms/step - loss: 0.8104 - accuracy: 0.7195 - val_loss: 0.9133 - val_accuracy: 0.6879
Epoch 4/10
391/391 [==============================] - 59s 152ms/step - loss: 0.7613 - accuracy: 0.7349 - val_loss: 0.9004 - val_accuracy: 0.6898
Epoch 5/10
391/391 [==============================] - 60s 154ms/step - loss: 0.7356 - accuracy: 0.7449 - val_loss: 0.8932 - val_accuracy: 0.6961
Epoch 6/10
391/391 [==============================] - 59s 151ms/step - loss: 0.6918 - accuracy: 0.7605 - val_loss: 0.8680 - val_accuracy: 0.7024
Epoch 7/10
391/391 [==============================] - 60s 154ms/step - loss: 0.6541 - accuracy: 0.7729 - val_loss: 0.8675 - val_accuracy: 0.7072
Epoch 8/10
391/391 [==============================] - 59s 151ms/step - loss: 0.6274 - accuracy: 0.7832 - val_loss: 0.9396 - val_accuracy: 0.6835
Epoch 9/10
391/391 [==============================] - 60s 154ms/step - loss: 0.6015 - accuracy: 0.7922 - val_loss: 0.8635 - val_accuracy: 0.7130
Epoch 10/10
391/391 [==============================] - 59s 151ms/step - loss: 0.5577 - accuracy: 0.8073 - val_loss: 0.8723 - val_accuracy: 0.7117
```



Set more arguments in convolutional layers and pool layers.

```
x_train, x_test = x_train / 255.0, x_test / 255.0

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), padding='valid', dilation_rate=(1,1), groups=1, use_bias=True,
                  kernel_initializer='glorot_uniform', bias_initializer='zeros', input_shape=(32, 32, 3)),
    layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid'),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=128)
```

```
                  kernel_initializer='glorot_uniform', bias_initializer='zeros'),
    layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid'),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=128)
```

```
Epoch 1/5
391/391 [==============================] - 51s 125ms/step - loss: 2.1330 - accuracy: 0.2020 - val_loss: 1.9402 - val_accuracy: 0.3052
Epoch 2/5
391/391 [==============================] - 49s 126ms/step - loss: 1.8673 - accuracy: 0.3268 - val_loss: 1.8150 - val_accuracy: 0.3461
Epoch 3/5
391/391 [==============================] - 48s 123ms/step - loss: 1.7632 - accuracy: 0.3628 - val_loss: 1.7076 - val_accuracy: 0.3803
Epoch 4/5
391/391 [==============================] - 49s 127ms/step - loss: 1.6863 - accuracy: 0.3883 - val_loss: 1.6339 - val_accuracy: 0.4075
Epoch 5/5
391/391 [==============================] - 49s 126ms/step - loss: 1.6293 - accuracy: 0.4088 - val_loss: 1.6092 - val_accuracy: 0.4157
<keras.callbacks.History at 0x7f3d0bdcebd0>
```

Use AveragePooling2D method.

```
x_train, x_test = x_train / 255.0, x_test / 255.0

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.AveragePooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), padding='valid', dilation_rate=(1,1), groups=1, use_bias=True,
                  kernel_initializer='glorot_uniform', bias_initializer='zeros'),
    layers.AveragePooling2D(pool_size=(2, 2), strides=None, padding='valid'),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.AveragePooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=128)
```

```
Epoch 1/5
391/391 [==============================] - 43s 109ms/step - loss: 2.3028 - accuracy: 0.0970 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 2/5
391/391 [==============================] - 44s 114ms/step - loss: 2.3027 - accuracy: 0.0962 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 3/5
391/391 [==============================] - 43s 110ms/step - loss: 2.3027 - accuracy: 0.0974 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 4/5
391/391 [==============================] - 43s 110ms/step - loss: 2.3027 - accuracy: 0.0977 - val_loss: 2.3026 - val_accuracy: 0.1000
Epoch 5/5
391/391 [==============================] - 45s 114ms/step - loss: 2.3027 - accuracy: 0.0992 - val_loss: 2.3026 - val_accuracy: 0.1000
<keras.callbacks.History at 0x7f3d0bc00b90>
```