alexnet-keras

0.1 Imports

[1]:
```
import numpy as np
import datetime
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar100
```

0.2 Preprocessing

0.2.1 Load dataset

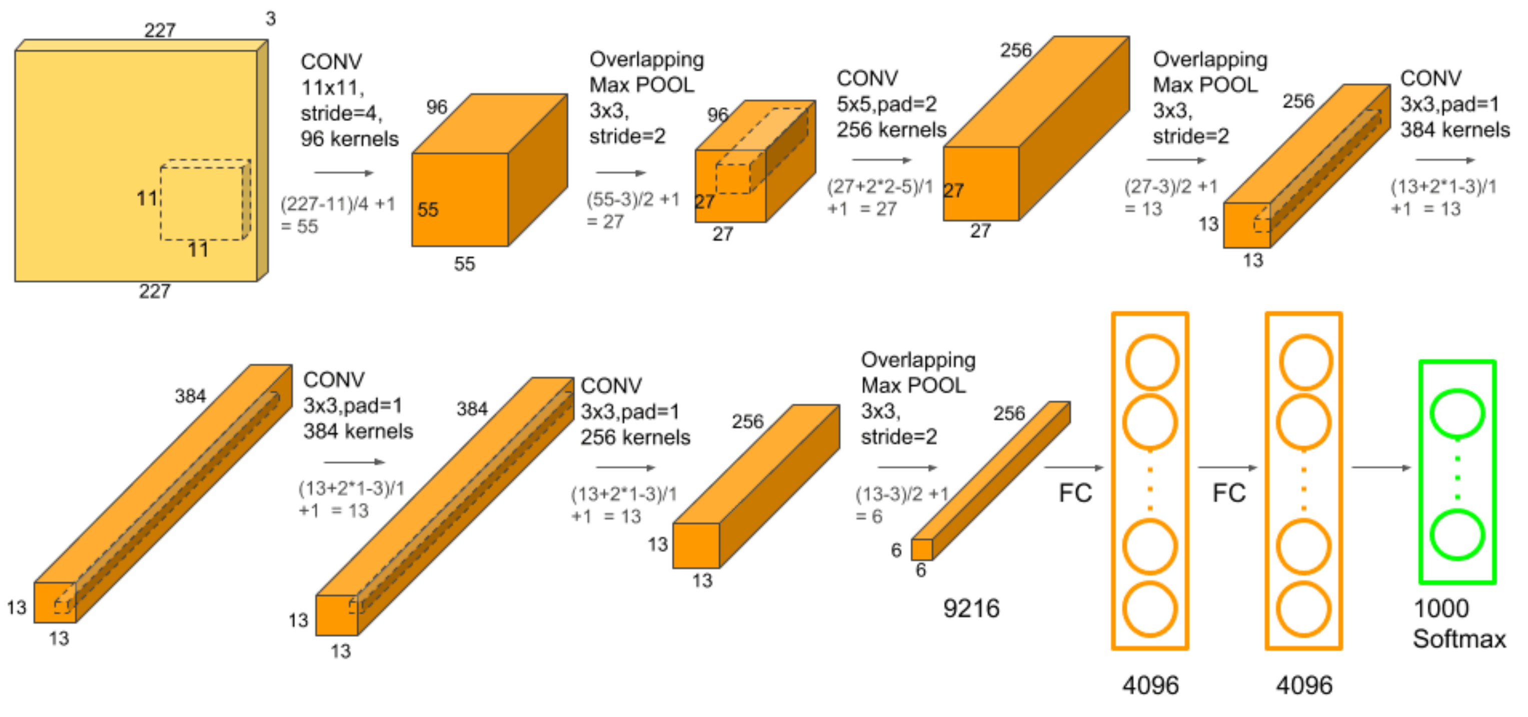
[2]:
```
(X_train, y_train), (X_test, y_test) = cifar100.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169009152/169001437 [==============================] - 6s 0us/step
```

0.2.2 Data normalization

[3]:
```
X_train = X_train / 255.0
X_test = X_test / 255.0

print(X_train.shape)
print(X_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

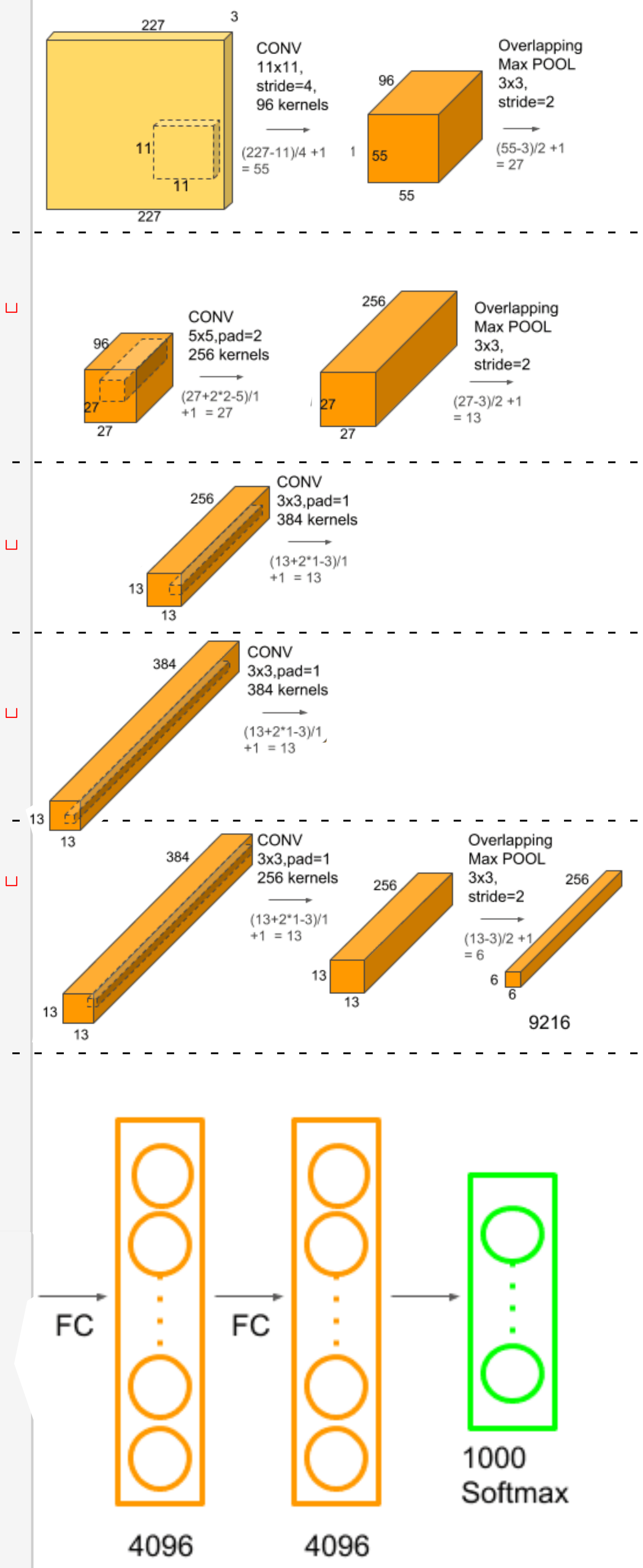


0.3 Learning

0.3.1 Building MLP

[4]:
```
model = tf.keras.models.Sequential()

# Input Layer
model.add(tf.keras.Input(shape=(32,32,3), ))
model.add(tf.keras.layers.experimental.preprocessing.Resizing(227,227))

# 1st Convolutional Layer
model.add(tf.keras.layers.Conv2D(filters=96, kernel_size=(11,11),␣
↪strides=(4,4), padding='valid'))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3), strides=(2,2),␣
↪padding='valid'))

# 2nd Convolutional Layer
model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1),␣
↪padding='same'))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3), strides=(2,2),␣
↪padding='valid'))

# 3rd Convolutional Layer
model.add(tf.keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),␣
↪padding='same'))
model.add(tf.keras.layers.Activation('relu'))

# 4th Convolutional Layer
model.add(tf.keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),␣
↪padding='same'))
model.add(tf.keras.layers.Activation('relu'))

# 5th Convolutional Layer
model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),␣
↪padding='same'))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3), strides=(2,2),␣
↪padding='valid'))

# 1 Fully Connected layer
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(4096, input_shape=(224*224*3,)))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Dropout(0.4))

# 2 Fully Connected Layer
model.add(tf.keras.layers.Dense(4096))
model.add(tf.keras.layers.Activation('relu'))
model.add(tf.keras.layers.Dropout(0.4))

# Output Layer
model.add(tf.keras.layers.Dense(100))
model.add(tf.keras.layers.Activation('relu'))

opt = tf.keras.optimizers.Adam()
model.compile(optimizer=opt, loss='sparse_categorical_crossentropy',␣
↪metrics=['sparse_categorical_accuracy'])

model.summary()
```

CIFAR dataset has 100 classes



```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resizing (Resizing)          (None, 227, 227, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 55, 55, 96)        34944
_________________________________________________________________
activation (Activation)      (None, 55, 55, 96)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 27, 256)       614656
_________________________________________________________________
activation_1 (Activation)    (None, 27, 27, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 384)       885120
_________________________________________________________________
activation_2 (Activation)    (None, 13, 13, 384)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 384)       1327488
_________________________________________________________________
activation_3 (Activation)    (None, 13, 13, 384)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 256)       884992
_________________________________________________________________
activation_4 (Activation)    (None, 13, 13, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              37752832
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               409700
_________________________________________________________________
activation_7 (Activation)    (None, 100)               0
=================================================================
Total params: 58,691,044
Trainable params: 58,691,044
Non-trainable params: 0

_________________________________________________________________
```

0.3.2 Train

[ ]:
```
history = model.fit(X_train, y_train, batch_size=100, validation_split=0.1,␣
↪epochs=5)
```

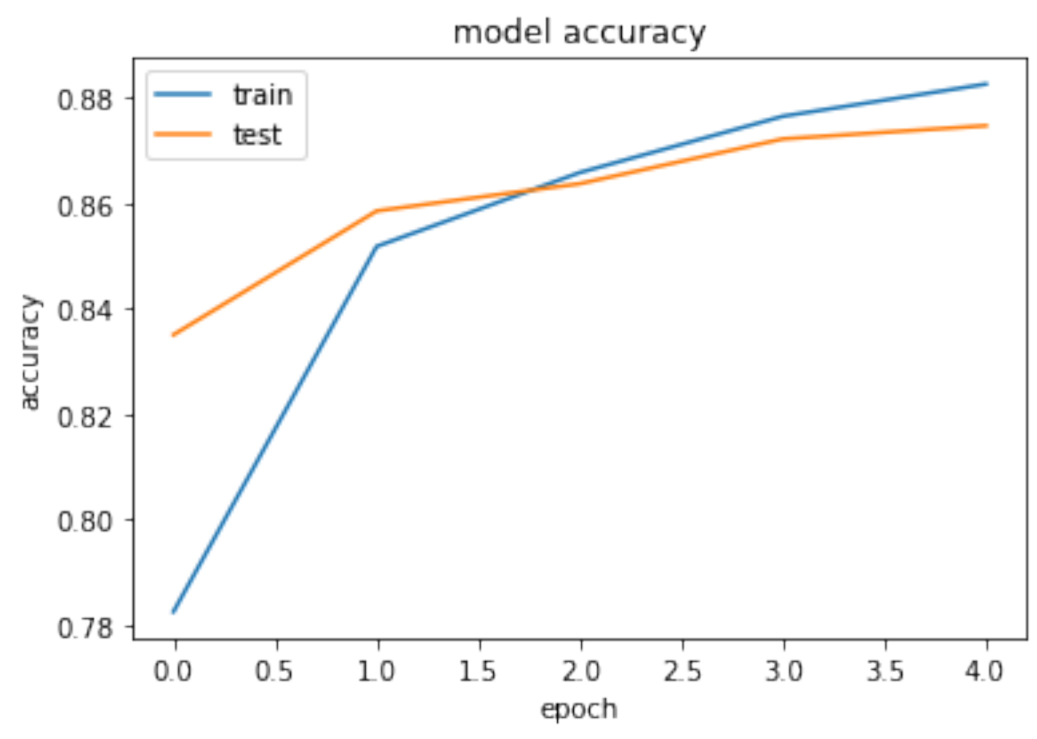
```
Epoch 1/5
```

0.3.3 Evaluation

[ ]:
```
test_loss, test_accuracy = model.evaluate(X_train, y_train, batch_size=100)
test_loss, test_accuracy = model.evaluate(X_test, y_test, batch_size=100)
```

```
500/500 [==============================] - 11s 22ms/step - loss: 4.6053 -
sparse_categorical_accuracy: 0.0100
100/100 [==============================] - 2s 21ms/step - loss: 4.6053 -
sparse_categorical_accuracy: 0.0100
```

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

plt.plot(history.history['sparse_categorical_accuracy'])
plt.plot(history.history['val_sparse_categorical_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```


model accuracy
train
test
accuracy
epoch