Assignment 6.1

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
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.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928     
_________________________________________________________________
flatten (Flatten)            (None, 576)               0         
_________________________________________________________________
dense (Dense)                (None, 64)                36928     
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650       
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
```

```
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 56s 58ms/step - loss: 0.1462 - accura
Epoch 2/5
938/938 [==============================] - 55s 58ms/step - loss: 0.0440 - accura
Epoch 3/5
938/938 [==============================] - 55s 59ms/step - loss: 0.0319 - accura
Epoch 4/5
938/938 [==============================] - 55s 59ms/step - loss: 0.0230 - accura
Epoch 5/5
938/938 [==============================] - 55s 58ms/step - loss: 0.0173 - accura
<keras.callbacks.History at 0x7ff1671e75d0>
```

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

```
313/313 [==============================] - 3s 10ms/step - loss: 0.0286 - accurac
```

```
test_acc
```

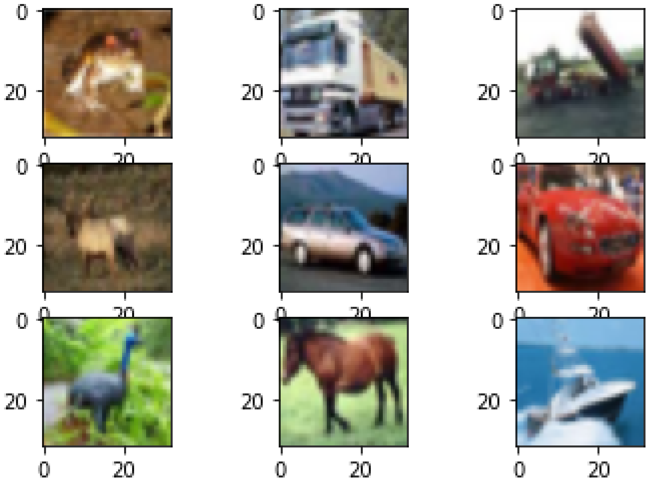
```
0.9925000071525574
```

Assignment 6.2 a.

```
from matplotlib import pyplot
from keras.datasets import cifar10
# load dataset
(trainX, trainy), (testX, testy) = cifar10.load_data()
# summarize loaded dataset
print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))
print('Test: X=%s, y=%s' % (testX.shape, testy.shape))
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
Train: X=(50000, 32, 32, 3), y=(50000, 1)
Test: X=(10000, 32, 32, 3), y=(10000, 1)
```

```
# plot first few images
for i in range(9):
    # define subplot
    pyplot.subplot(330 + 1 + i)
    # plot raw pixel data
    pyplot.imshow(trainX[i])
# show the figure
pyplot.show()
```



```
from tensorflow.keras.utils import to_categorical
# one hot encode target values
trainY = to_categorical(trainy)
testY = to_categorical(testy)
```

```
# from keras import layers
# from keras import models

# 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(128, (3, 3), activation='relu'))
# model.add(layers.MaxPooling2D((2, 2)))
# model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# model.add(layers.MaxPooling2D((2, 2)))
# model.add(layers.Flatten())
# model.add(layers.Dense(512, activation='relu'))
# model.add(layers.Dense(1, activation='softmax')) # sigmoid
```

```


# example of a 3-block vgg style architecture
from keras import layers
from keras import models
```

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform
model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform
model.add(layers.MaxPooling2D((2, 2)))
```

```
# example output part of the model
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(layers.Dense(10, activation='softmax'))
```

```
# compile model
from keras import optimizers
opt = optimizers.RMSprop(lr=1e-4)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/opti
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

```
# from keras import optimizers

# model.compile(loss='categorical_crossentropy',#'binary_crossentropy',
#               optimizer=optimizers.RMSprop(lr=1e-4),
#               metrics=['acc'])
```

```
# history = model.fit_generator(
#       train_generator,
#       steps_per_epoch=100,
#       epochs=30,
#       validation_data=validation_generator,
#       validation_steps=50)
```

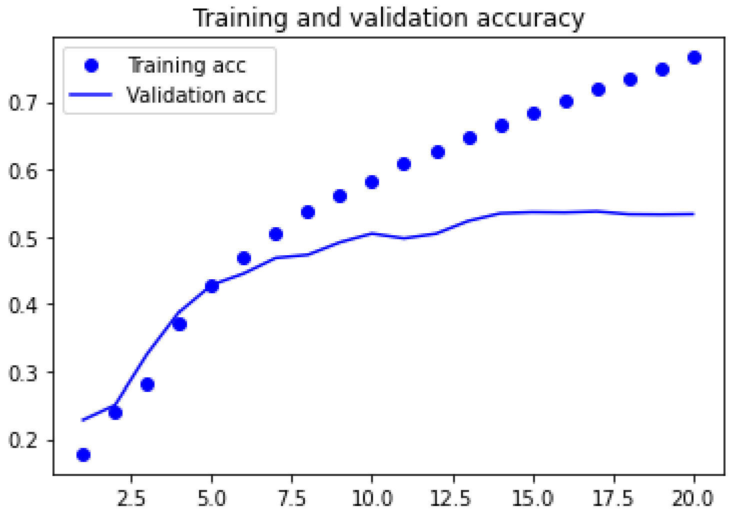
```
history = model.fit(
      trainX,trainY,
      steps_per_epoch=100,
      epochs=20,
      validation_data=(testX, testY),
      validation_steps=50)
```

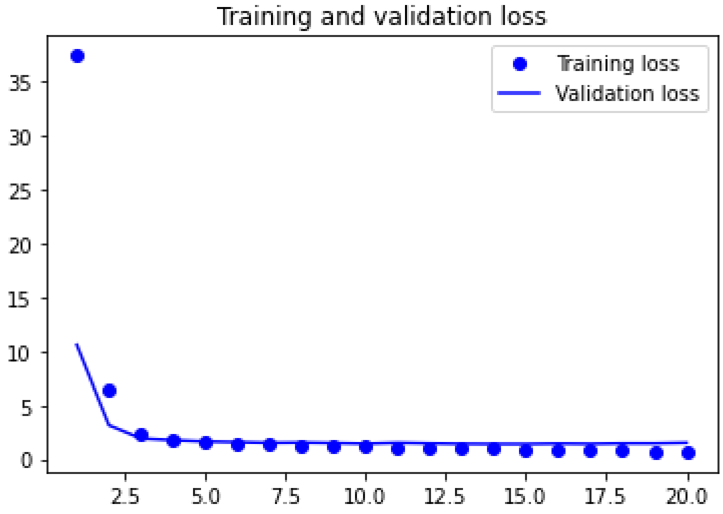
```
Epoch 1/20
100/100 [==============================] - 355s 4s/step - loss: 75.4307 - accura
Epoch 2/20
```

```
100/100 [==============================] - 352s 4s/step - loss: 8.6033 - accurac
Epoch 3/20
100/100 [==============================] - 353s 4s/step - loss: 2.6168 - accurac
Epoch 4/20
100/100 [==============================] - 351s 4s/step - loss: 1.8304 - accurac
Epoch 5/20
100/100 [==============================] - 352s 4s/step - loss: 1.6397 - accurac
Epoch 6/20
100/100 [==============================] - 351s 4s/step - loss: 1.5048 - accurac
Epoch 7/20
100/100 [==============================] - 352s 4s/step - loss: 1.3915 - accurac
Epoch 8/20
100/100 [==============================] - 351s 4s/step - loss: 1.3159 - accurac
Epoch 9/20
100/100 [==============================] - 351s 4s/step - loss: 1.2402 - accurac
Epoch 10/20
100/100 [==============================] - 348s 3s/step - loss: 1.1812 - accurac
Epoch 11/20
100/100 [==============================] - 348s 3s/step - loss: 1.1228 - accurac
Epoch 12/20
100/100 [==============================] - 348s 3s/step - loss: 1.0467 - accurac
Epoch 13/20
100/100 [==============================] - 349s 3s/step - loss: 1.0122 - accurac
Epoch 14/20
100/100 [==============================] - 348s 3s/step - loss: 0.9552 - accurac
Epoch 15/20
100/100 [==============================] - 348s 3s/step - loss: 0.9008 - accurac
Epoch 16/20
100/100 [==============================] - 349s 3s/step - loss: 0.8524 - accurac
Epoch 17/20
100/100 [==============================] - 357s 4s/step - loss: 0.7881 - accurac
Epoch 18/20
100/100 [==============================] - 340s 3s/step - loss: 0.7648 - accurac
Epoch 19/20
100/100 [==============================] - 342s 3s/step - loss: 0.7155 - accurac
Epoch 20/20
100/100 [==============================] - 341s 3s/step - loss: 0.6649 - accurac
```

```
model.save('cifar10.h5')
```

```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





Assignment 6.2 b.

```
# from keras.preprocessing import image
# from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# create data generator
# datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1, horizont
datagen = ImageDataGenerator(
      rotation range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

# prepare iterator
it_train = datagen.flow(trainX, trainY, batch_size=64)
```

```


# example of a 3-block vgg style architecture
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniforn
model.add(layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniforn
model.add(layers.MaxPooling2D((2, 2)))

# example output part of the model
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(layers.Dense(10, activation='softmax'))

# compile model
from keras import optimizers
opt = optimizers.RMSprop(lr=1e-4)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

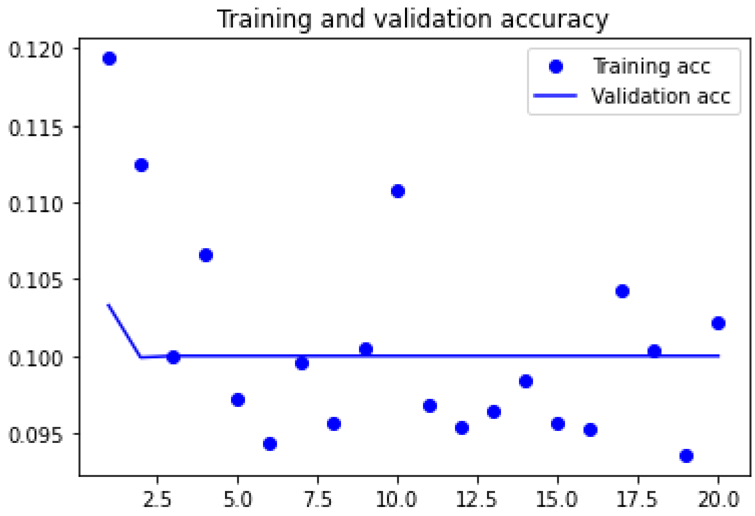
```
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/optin
      "The `lr` argument is deprecated, use `learning_rate` instead.")
```

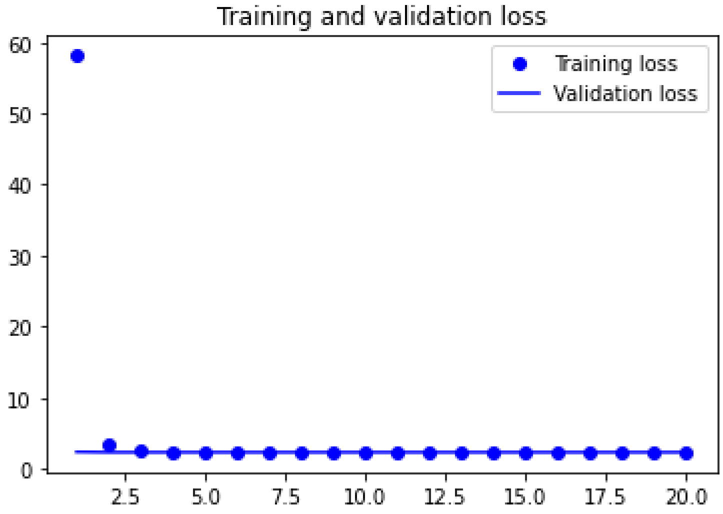
```
# history = model.fit(trainX, trainY, epochs=40, validation_data=(testX, testY), batch
# # fit model
# steps = int(trainX.shape[0] / 64)
# history = model.fit_generator(it_train, steps_per_epoch=steps, epochs=100, validatic
```

```
history = model.fit_generator(
      it_train,
      steps_per_epoch=100,
      epochs=20,
      validation_data=(testX, testY)
      validation_data=(testX, testY),
      validation_steps=50)
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarnin
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/20
100/100 [==============================] - 64s 623ms/step - loss: 132.1379 - accu
Epoch 2/20
100/100 [==============================] - 62s 617ms/step - loss: 4.4708 - accura
Epoch 3/20
100/100 [==============================] - 62s 620ms/step - loss: 2.4597 - accura
Epoch 4/20
100/100 [==============================] - 61s 612ms/step - loss: 2.3638 - accura
Epoch 5/20
100/100 [==============================] - 61s 611ms/step - loss: 2.3323 - accura
Epoch 6/20
100/100 [==============================] - 61s 613ms/step - loss: 2.3304 - accura
Epoch 7/20
100/100 [==============================] - 61s 613ms/step - loss: 2.3211 - accura
Epoch 8/20
100/100 [==============================] - 61s 610ms/step - loss: 2.3194 - accura
Epoch 9/20
100/100 [==============================] - 61s 616ms/step - loss: 2.3292 - accura
Epoch 10/20
100/100 [==============================] - 61s 615ms/step - loss: 2.3183 - accura
Epoch 11/20
100/100 [==============================] - 62s 619ms/step - loss: 2.3107 - accura
Epoch 12/20
100/100 [==============================] - 61s 616ms/step - loss: 2.3139 - accura
Epoch 13/20
100/100 [==============================] - 63s 627ms/step - loss: 2.3110 - accura
Epoch 14/20
100/100 [==============================] - 63s 628ms/step - loss: 2.3095 - accura
Epoch 15/20
100/100 [==============================] - 63s 634ms/step - loss: 2.3122 - accura
Epoch 16/20
100/100 [==============================] - 62s 620ms/step - loss: 2.3084 - accura
Epoch 17/20
100/100 [==============================] - 59s 592ms/step - loss: 2.3104 - accura
Epoch 18/20
100/100 [==============================] - 57s 575ms/step - loss: 2.3150 - accura
Epoch 19/20
100/100 [==============================] - 60s 600ms/step - loss: 2.3042 - accura
Epoch 20/20
100/100 [==============================] - 64s 639ms/step - loss: 2.3151 - accura
```

```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





✓ 0s completed at 6:39 PM ● ×