UVA CS 4774

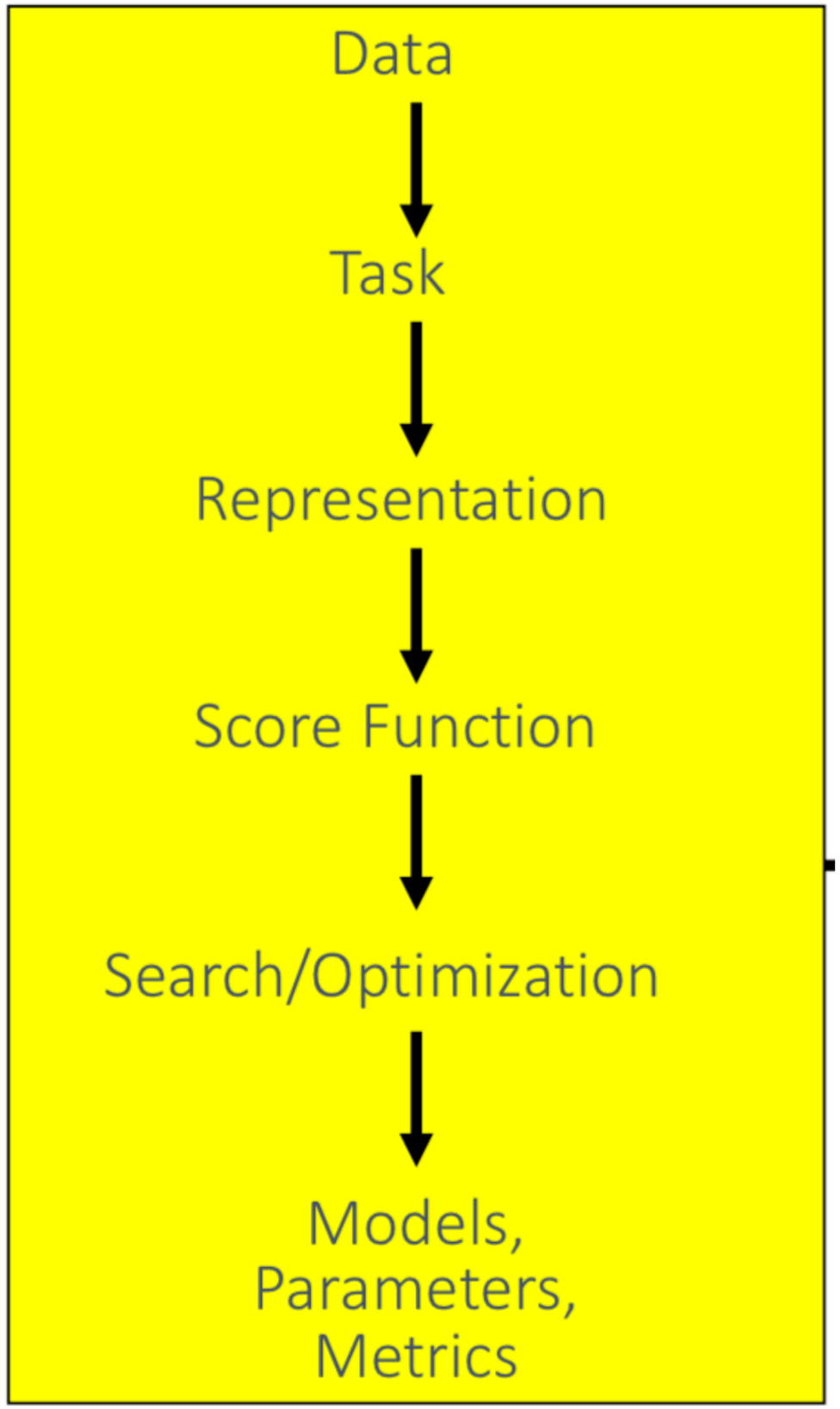
A Brief Introduction to Keras

Presented By: Zhe Wang

Professor: Dr. Yanjun Qi

October 13, 2020

Nutshell for the Deep Learning



I: Data

Keras.datasets module provide several toy datasets including MNIST, CIFAR10, CIFAR100, etc.

First load the dataset:

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data(path="mnist.npz")
```

Preprocessing:

```
x_train = x_train.astype("float32") / 255  # Scale pixel values to [0, 1]
x_train = np.expand_dims(x_train, -1)   # Size of x_train = [60000, 28, 28, 1], for CNN.
x_train = x_train.reshape(-1, 784)  # Size of x_train = [60000, 784], for MLP.
```

```
y_train = keras.utils.to_categorical(y_train, num_classes) # (60000, ) to (60000, 10)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

II: Model

Two ways to build your model: sequential and functionals

Sequential:

```
input_shape = (28, 28, 1)
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="sigmoid"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, 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
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
```

Functional:

```
class MyModel(keras.Model):

  def __init__(self):
    super(MyModel, self).__init__()
    self.dense1 = keras.layers.Dense(4, activation='relu')
  def call(self, inputs):
    x = self.dense1(inputs)
    return x

model = MyModel()
```

Print out your model:

```
model = MyModel()
model.build(input_shape=(None,784))
model.summary()
```

```
Model: "my_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              multiple                  3140
=================================================================
Total params: 3140
Trainable params: 3140
Non-trainable params: 0
_________________________________________________________________
```

II: Define your optimizer

```
my_opt = keras.optimizers.Adam(learning_rate=1e-3)  # Adam, SGD, Adagrad, RMSprop, etc.
```

III: Define your loss function

```
my_loss = keras.losses.CategoricalCrossentropy(
    label_smoothing=0,
    name="categorical_crossentropy",
)
```

IV: Compile your model: configures the model for training

```
model.compile(loss=my_loss, optimizer=my_opt, metrics=["accuracy"])
```

Or, simply calling:

```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

VI: Start your training

```
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1, shuffle=True)
```

```
Epoch 1/5
422/422 [==============================] - 48s 114ms/step - loss: 0.3594 - accuracy: 0.8901 - val_loss: 0.0810 - val_accuracy: 0.9772
Epoch 2/5
422/422 [==============================] - 42s 100ms/step - loss: 0.1094 - accuracy: 0.9663 - val_loss: 0.0552 - val_accuracy: 0.9845
Epoch 3/5
422/422 [==============================] - 33s 78ms/step - loss: 0.0824 - accuracy: 0.9742 - val_loss: 0.0508 - val_accuracy: 0.9860
Epoch 4/5
422/422 [==============================] - 33s 78ms/step - loss: 0.0691 - accuracy: 0.9789 - val_loss: 0.0446 - val_accuracy: 0.9880
Epoch 5/5
422/422 [==============================] - 30s 72ms/step - loss: 0.0593 - accuracy: 0.9815 - val_loss: 0.0429 - val_accuracy: 0.9892
```

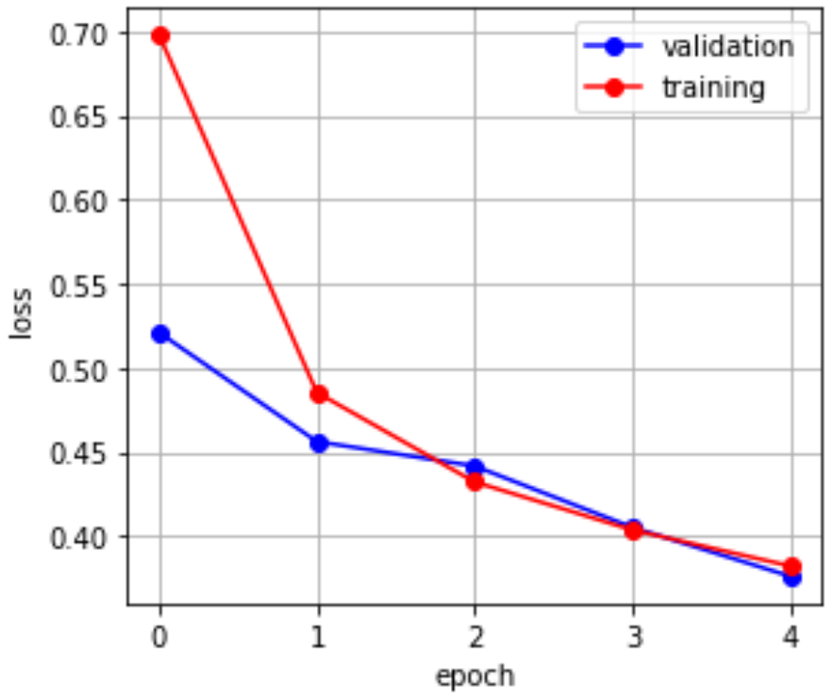
VII: Start your evaluation

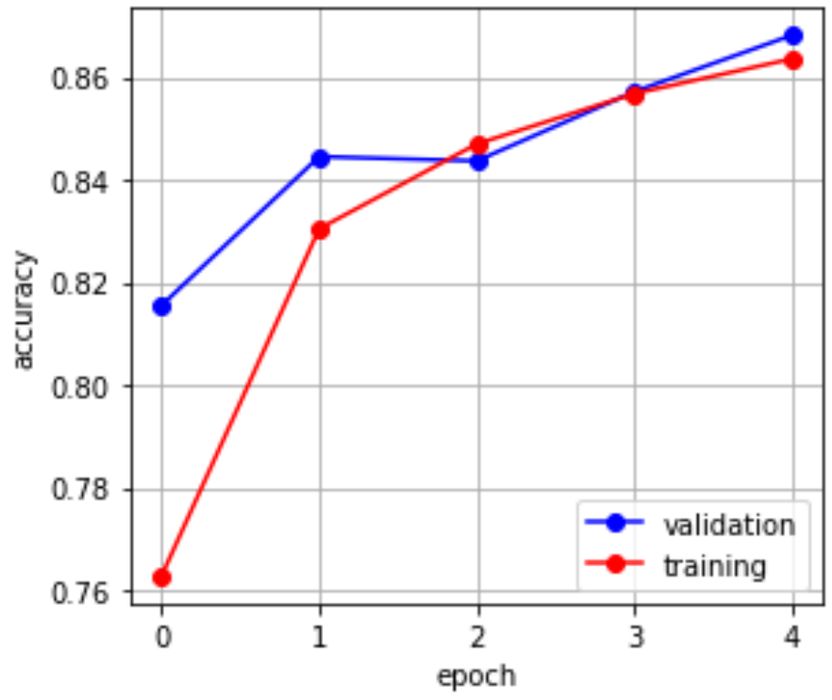
```
model.evaluate(x_test, y_test)  # Returns the loss value & metrics values for the model in test mode.
```

VIII: Start your analysis and visualize your metric

```
train_loss_history = history.history['loss']
val_loss_history = history.history['val_loss']

train_acc_history = history.history['accuracy']
val_acc_history = history.history['val_accuracy']
```





Extra:

1: Custom data generator: https://www.tensorflow.org/api_docs/python/tf/keras/utils/Sequence

2: Data augmentation: https://keras.io/api/preprocessing/image/

3: Custom layer: https://keras.io/guides/making_new_layers_and_models_via_subclassing/

4: Custom metric: https://github.com/borundev/ml_cookbook/blob/master/Custom%20Metric%20(Confusion%20Matrix)%20and%20train_step%20method.ipynb