機器學習於材料資訊的應用
Machine Learning on Material Informatics

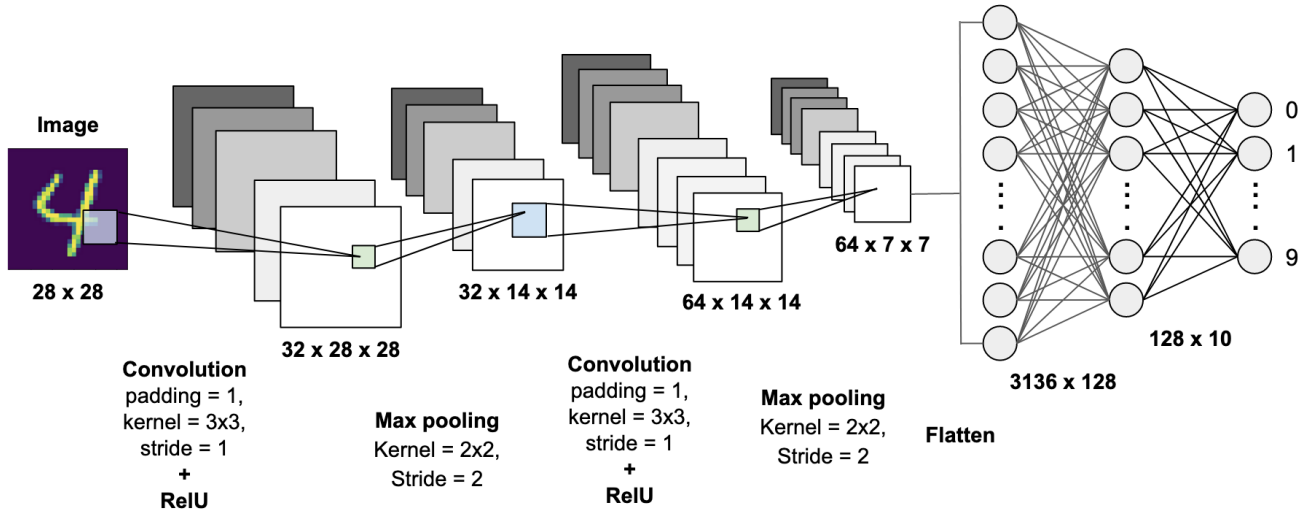
陳南佑(NAN-YOW CHEN)

nanyow@narlabs.org.tw

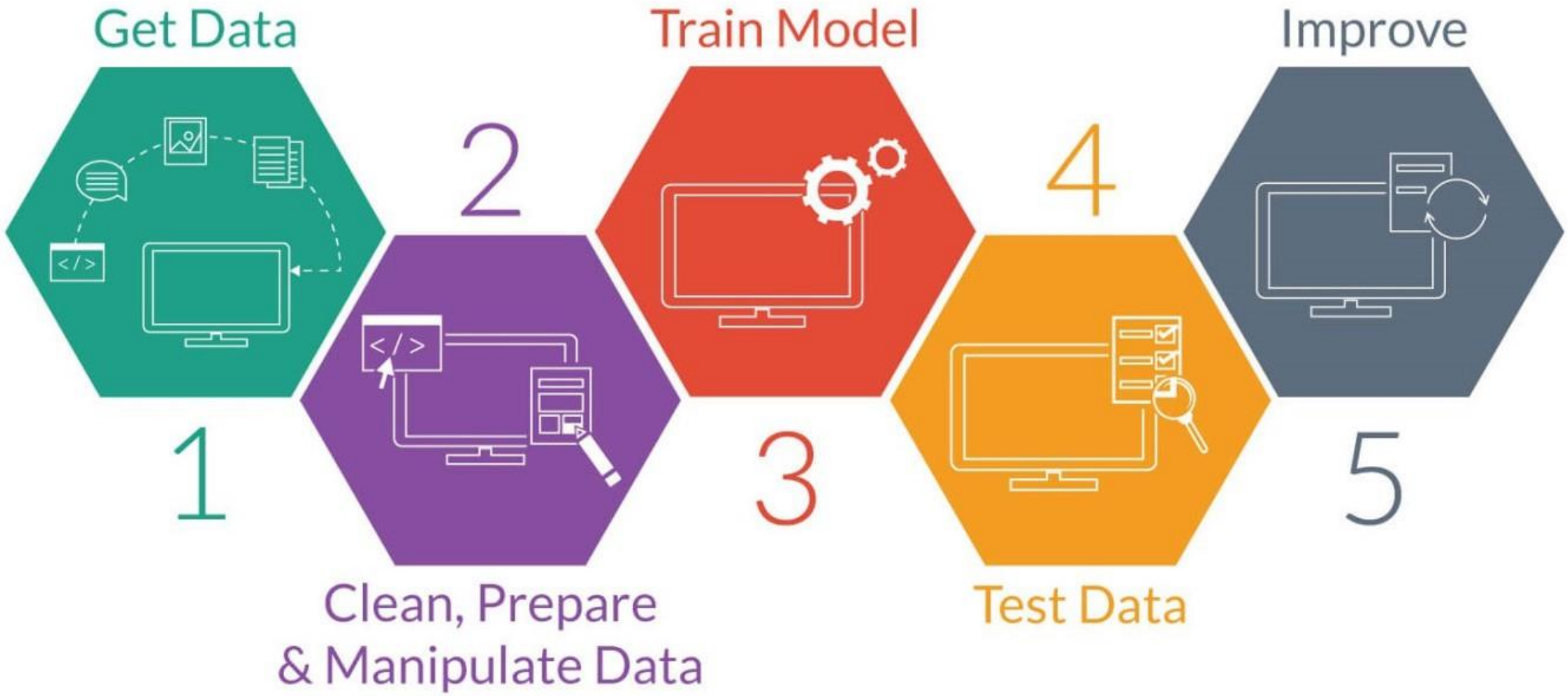
楊安正(AN-CHENG YANG)

acyang@narlabs.org.tw

Handwritten Digits classification by CNN



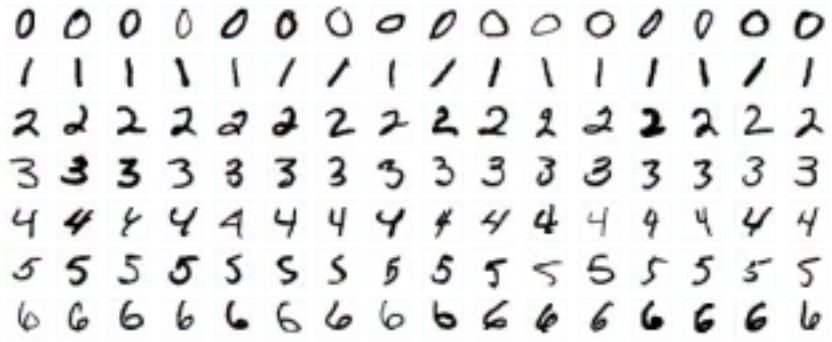
http://yann.lecun.com/exdb/mnist/


Get Data
1
2
Clean, Prepare
& Manipulate Data
Train Model
3
4
Test Data
Improve
5

Get Data

THE MNIST DATABASE

of handwritten digits



MNIST database 由兩種資料來源組成NIST‘s Special Database 3(SD-3)和 Special Database 1(SD-1)。
SD-3 品質比SD-1更乾淨更容易分類。

http://yann.lecun.com/exdb/mnist/

1. 手動下載 wget curl
2. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/input_data.py (即將廢棄)
3. https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data
4. …

Clean, Prepare, Manipulate Data

Import Data, One-Hot Encoding

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./data/", one_hot=True)
```

取用train set

```
mnist.train.images
mnist.train.labels
```

取用test set

```
mnist.test.images
mnist.test.labels
```

取用validation set

```
mnist.validation.images
mnist.validation.labels
```

	SD-3	SD-1
Training Dataset	27500	27500
Test Dataset	5000	5000
Validation data	2500	2500

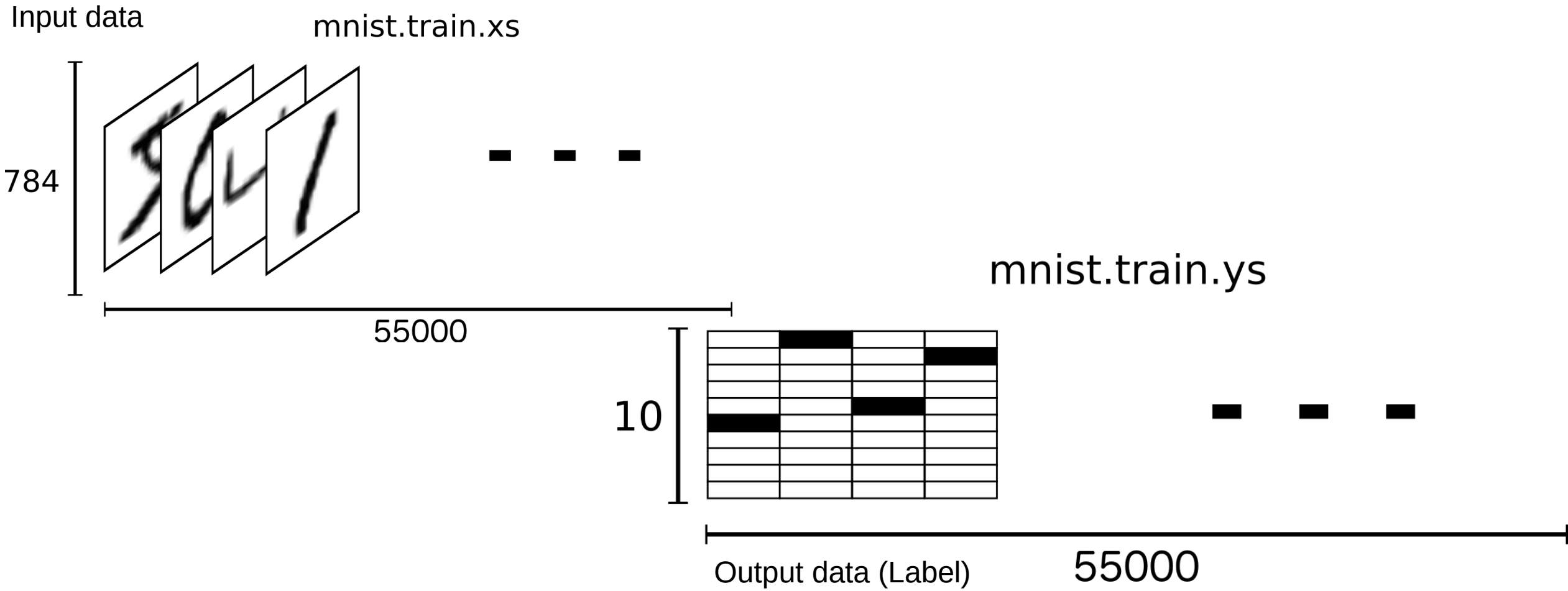
Clean, Prepare, Manipulate Data

Output data (Label)

- 每張圖的答案(Label)就是數字本身，例如0,1,2,…,9，不做特別處理的編碼稱為自然狀態碼。
- One-Hot Encoding又稱一位有效編碼，其方法是使用N位狀態暫存器來對N個狀態進行編碼，每個狀態都有它獨立的暫存器位，並且在任意時候，其中只有一位有效。例如：

自然狀態碼	一位有效編碼
0	[1,0,0,0,0,0,0,0,0,0]
1	[0,1,0,0,0,0,0,0,0,0]
…	…
9	[0,0,0,0,0,0,0,0,0,1]

Clean, Prepare, Manipulate Data



Train Model

Setting training parameter

```
# Parameters
learning_rate = 0.001
num_steps = 500
batch_size = 50
display_step = 100
```

Setting random seed for tf & np
(For debug)

```
tf.set_random_seed(1)
np.random.seed(2)
```

Define input & output

```
# https://www.tensorflow.org/api_docs/python/tf/placeholder
tf_x = tf.placeholder(tf.float32, [None, 28*28])       # input
tf_y = tf.placeholder(tf.float32, [None, 10])     # label

image = tf.reshape(tf_x, [-1, 28, 28, 1])   # (batch, height, width, channel)
```

Train Model

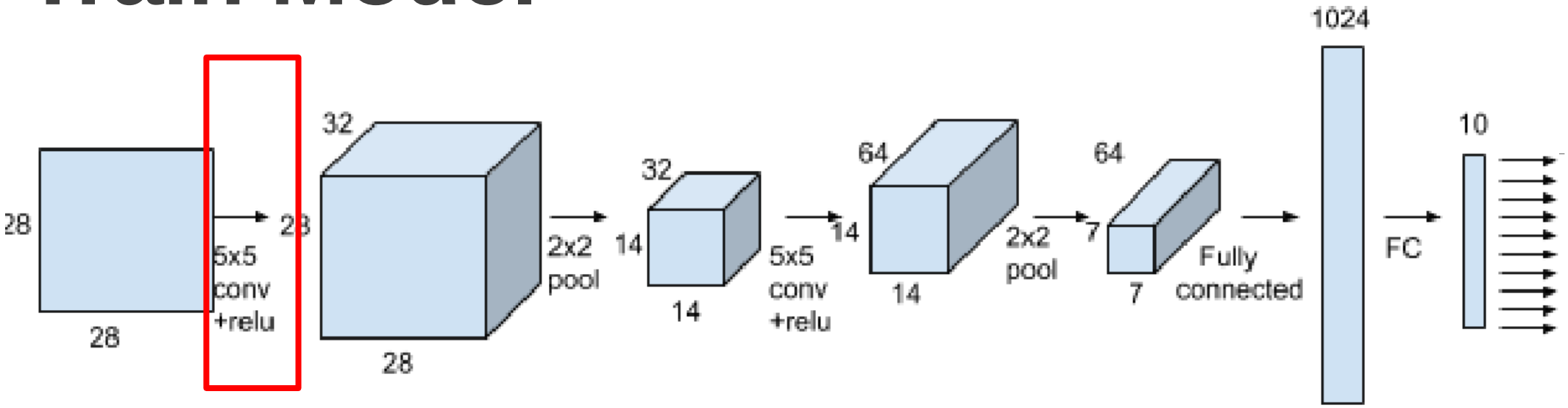


Figure D.2: Network architecture for MNIST classifier CNN

2D convolution layer

```
conv1 = tf.layers.conv2d(
    inputs=image,
    filters=32,
    kernel_size=[5, 5],
    padding='same',
    activation=tf.nn.relu)            # -> (28, 28, 32)

print("shape of conv1 is", conv1.shape)
```

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/layers/Conv2D

tf.layers.Conv2D(args, …)

- **Arguments:**
  - inputs: Tensor input.
  - filters: Integer, the number of filters in the convolution.
  - kernel_size: An tuple/list of 2 integers, specifying the height and width of the 2D convolution window.
  - padding: One of "valid" or "same" (case-insensitive).
  - strides: An tuple/list of 2 integers, specifying the strides of the convolution along the height and width.
  - data_format: A string, one of channels_last (default) or channels_first.
    - channels_last corresponds to inputs with shape (batch, height, width, channels)
    - channels_first corresponds to inputs with shape (batch, channels, height, width).
  - activation: Activation function. Set it to None to maintain a linear activation.
  - name: A string, the name of the layer.

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/layers/Conv2D

Train Model

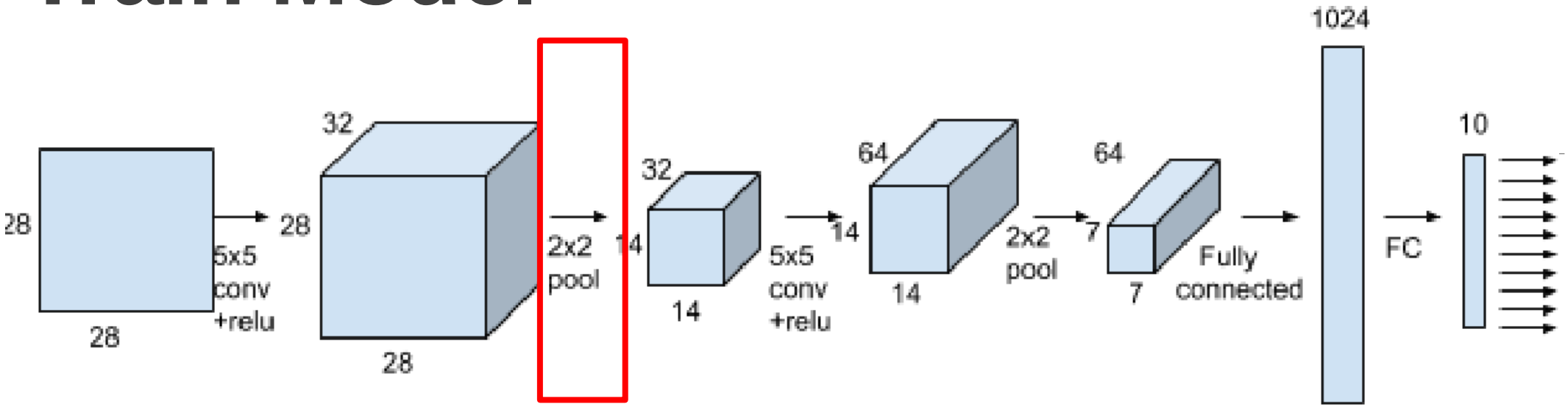


Figure D.2: Network architecture for MNIST classifier CNN

Max pooling layer for 2D inputs

```
pool1 = tf.layers.max_pooling2d(
    inputs=conv1,
    pool_size=[2, 2],
    strides=2)              # -> (14, 14, 32)

print("shape of pool1 is", pool1.shape)
```

tf.layers.MaxPooling2D(args, …)

- **Arguments:**
  - inputs: Tensor input.
  - pool_size: An tuple/list of 2 integers: (pool_height, pool_width) specifying the size of the pooling window.
  - padding: One of "valid" or "same" (case-insensitive).
  - strides: An tuple/list of 2 integers, specifying the strides of the pooling operation.
  - data_format: A string, one of channels_last (default) or channels_first.
    - channels_last corresponds to inputs with shape (batch, height, width, channels)
    - channels_first corresponds to inputs with shape (batch, channels, height, width).
  - name: A string, the name of the layer.

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/layers/MaxPooling2D

Train Model

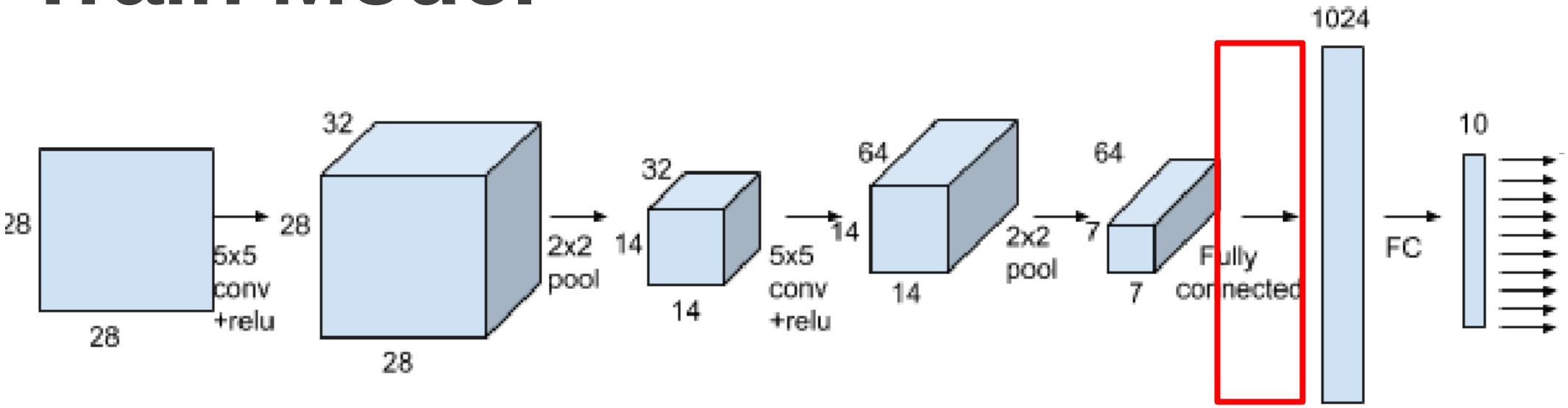


Figure D.2: Network architecture for MNIST classifier CNN

Flattens an input tensor

```
flat = tf.layers.Flatten()(pool2)
print("shape of flat is", flat.shape)

#flat = tf.reshape(pool2, [-1, 7*7*64])          # -> (7*7*64, )
#print("shape of flat is", flat.shape)
```

tf.layers.Flatten(args, …)

- **Arguments:**
  - data_format: A string, one of channels_last (default) or channels_first.
    - channels_last corresponds to inputs with shape (batch, height, width, channels)
    - channels_first corresponds to inputs with shape (batch, channels, height, width).

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/layers/Flatten

tf.layers.Dropout(args, …)

- **Arguments:**
  - inputs: Tensor input.
  - rate: The dropout rate, between 0 and 1. E.g. rate=0.1 would drop out 10% of input units.
  - name: The name of the layer (string).

Train Model

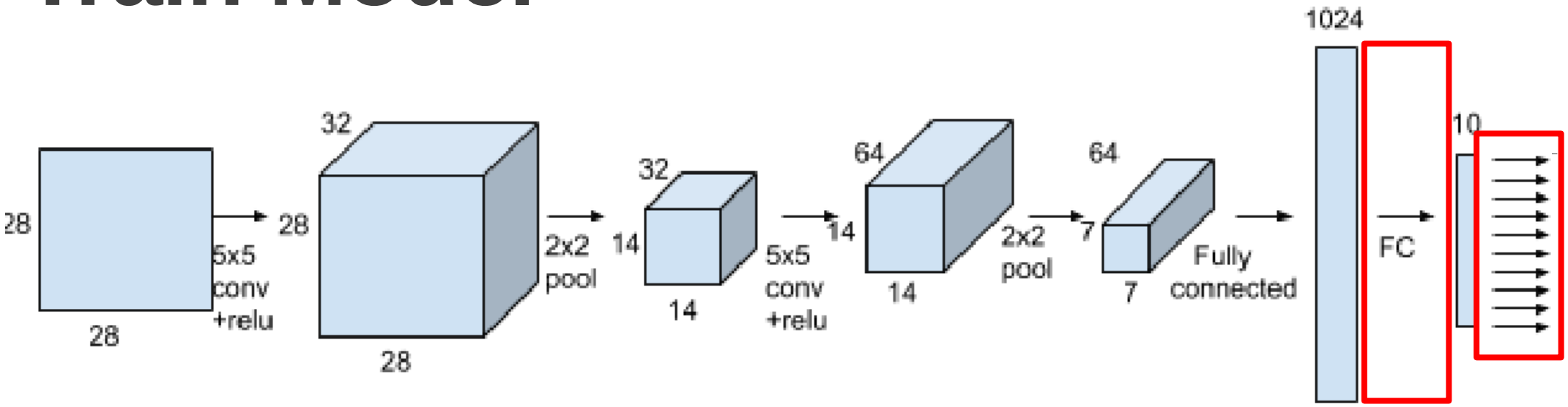


Figure D.2: Network architecture for MNIST classifier CNN

Densely-connected layer

```
dense2 = tf.layers.dense(
    inputs=flat,
    units=1024,
    activation=tf.nn.relu)
```

```
output = tf.layers.dense(
    inputs=dense2,
    units=10)                # output layer
```

Train Model

```
# Define loss and optimizer

#https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits

loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=output,
labels=tf_y))

optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)

train_op = optimizer.minimize(loss_op)
```

Train Model

Evaluate model (自行定義)

```
# https://www.tensorflow.org/api_docs/python/tf/math/argmax
correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y, 1))
# https://www.tensorflow.org/api_docs/python/tf/dtypes/cast
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

`tf.argmax:` 找出最大值的位置
tf.equal : 比較是否相等，回傳A Tensor of type bool with the same size as that of x or y.
tf.cast : 將bool轉成tf.float32
tf.reduce_mean : Computes the mean of elements

Evaluate model (使用tf定義)

```
accuracy = tf.metrics.accuracy(labels=tf.argmax(tf_y, axis=1),
predictions=tf.argmax(output, axis=1))[1]
```

tf.metrics.accuracy(args, …)

- **Arguments:**
  - labels: The ground truth values, a Tensor whose shape matches predictions.
  - predictions: The predicted values, a Tensor of any shape.
  - name: An optional variable_scope name.
- The accuracy function creates two local variables, total and count that are used to compute the frequency with which predictions matches labels.
- This frequency is ultimately returned as accuracy: an idempotent operation that simply divides total by count.

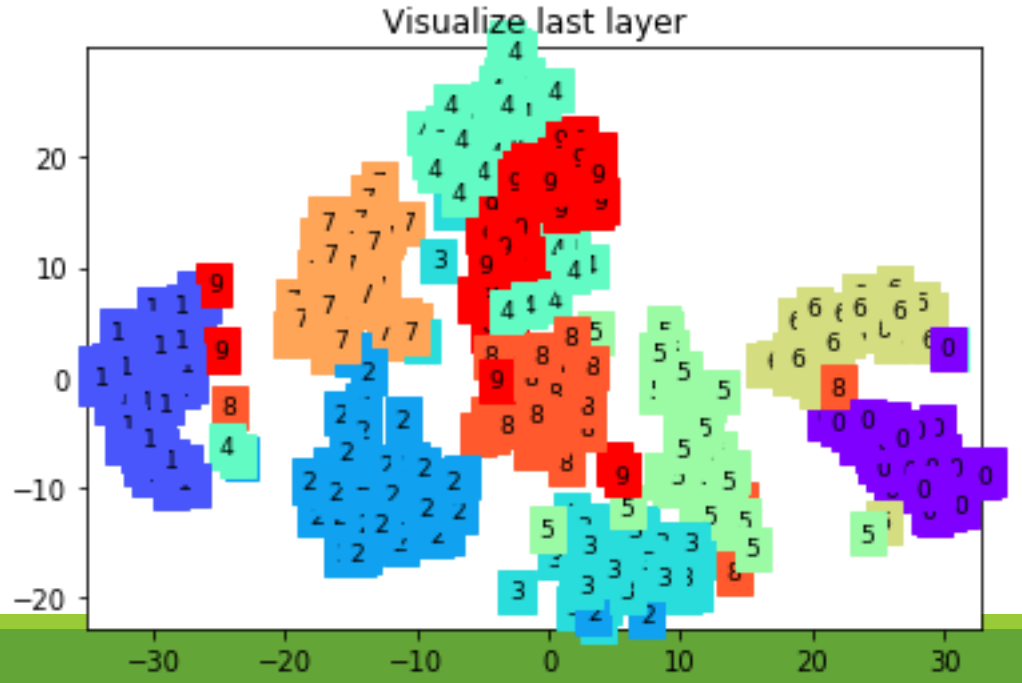
https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/metrics/accuracy

Visualization t-SNE

```
from matplotlib import cm
try: from sklearn.manifold import TSNE; HAS_SK = True
except: HAS_SK = False; print('\nPlease install sklearn for layer visualization\n')
def plot_with_labels(lowDWeights, labels):
    plt.cla(); X, Y = lowDWeights[:, 0], lowDWeights[:, 1]
    for x, y, s in zip(X, Y, labels):
        c = cm.rainbow(int(255 * s / 9)); plt.text(x, y, s, backgroundcolor=c,
fontsize=9)
    plt.xlim(X.min(), X.max()); plt.ylim(Y.min(), Y.max()); plt.title('Visualize
last layer'); plt.show(); plt.pause(0.01)
```

t-SNE

- t-SNE（t-distributed stochastic neighbor embedding，t-隨機鄰近嵌入法）是一種非線性的機器學習降維方法，與經典的 PCA相比，t-SNE 降維時保持局部結構的能力十分傑出，因此成為近年來學術論文與模型比賽中資料視覺化的常客。
- t-SNE 常用來將資料投影到 2 維或 3 維的空間作定性的視覺化觀察，通過視覺化直觀的驗證某資料集或演算法的有效性。



Train Model

```
# https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/group

init_op = tf.group(tf.global_variables_initializer(),
tf.local_variables_initializer()) # the local var is for accuracy_op


# 'Saver' op to save and restore all the variables

saver = tf.train.Saver()
```

tf.group(args, …)

- Create an op that groups multiple operations.
- **Arguments:**
  - *inputs: Zero or more tensors to group.
  - name: A name for this operation (optional).

https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/group

Train Model

```
batch_x, batch_y = mnist.train.next_batch(batch_size)
# Run optimization op (backprop)
sess.run(train_op, feed_dict={X: batch_x, Y: batch_y})
```

- Batch_Size 是機器學習中一個重要的參數，Batch 的選擇會決定梯度下降的方向。
- 如果dataset比較小，那麼可以採用full dataset的方式。優點就是full dataset的方向更能代表母體，可以準確地找到極值方向。
- 另外一個極端是一次只載入一個數據。每個樣本的修正方向以各自樣本的梯度方向修正，批次愈小，對於方向的估計愈不準確。
- 找一個適中的 Batch_Size 值就很重要。
- 如果dataset夠多，那麼用一半的data算出來的梯度與用full dataset幾乎一樣的。在合理範圍內，增大 Batch_Size ，(類似convergence test)。

Train Model

```
# Start training
with tf.Session() as sess:
    sess.run(init_op)     # initialize var in graph
    for step in range(num_steps):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        _, loss_ = sess.run([train_op, loss_op], {tf_x: batch_x, tf_y: batch_y})
        if step % display_step == 0:
            accuracy_, flat_representation = sess.run([accuracy, flat], {tf_x: test_x,
tf_y: test_y})
            print('Step:', step, '| train loss: %.4f' % loss_, '| test accuracy: %.2f' %
accuracy_)
```

Use Model

```
# Evaluate model
prediction = tf.nn.softmax(output)
ans = tf.argmax(prediction, 1)

# Running a test dataset by loading the model saved earlier
with tf.Session() as sess:
    # Run the initializer
    sess.run(init_op)
    saver.restore(sess, "cnn_model_old")
    print("Model restored from file: %s" % save_path)
    # Calculate the answer for the image
    print("Answer:", sess.run(ans, feed_dict={tf_x: mnist.test.images[0:1]}))
    print("prediction:", sess.run(prediction, feed_dict={tf_x: mnist.test.images[0:1]}))
```

使用tensorflow的keras api來讀資料和網路

Get Data

```
# https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)
```

注意這裡的x_train…都是numpy array，keras會幫你處理產生tensor的部分。

Train Model

Configures the model for training

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

Trains the model for a fixed number of epochs

```
model.fit(X_train, y_train_oh, epochs=1, batch_size=64)
```

Test Model

Returns the loss value & metrics values for the model in test mode.

```
model.evaluate(X_test, y_test_oh)
```

Use the trained model to predict

```
model.predict(X_test[0].reshape((1, 1, 28, 28)), batch_size=1)
```

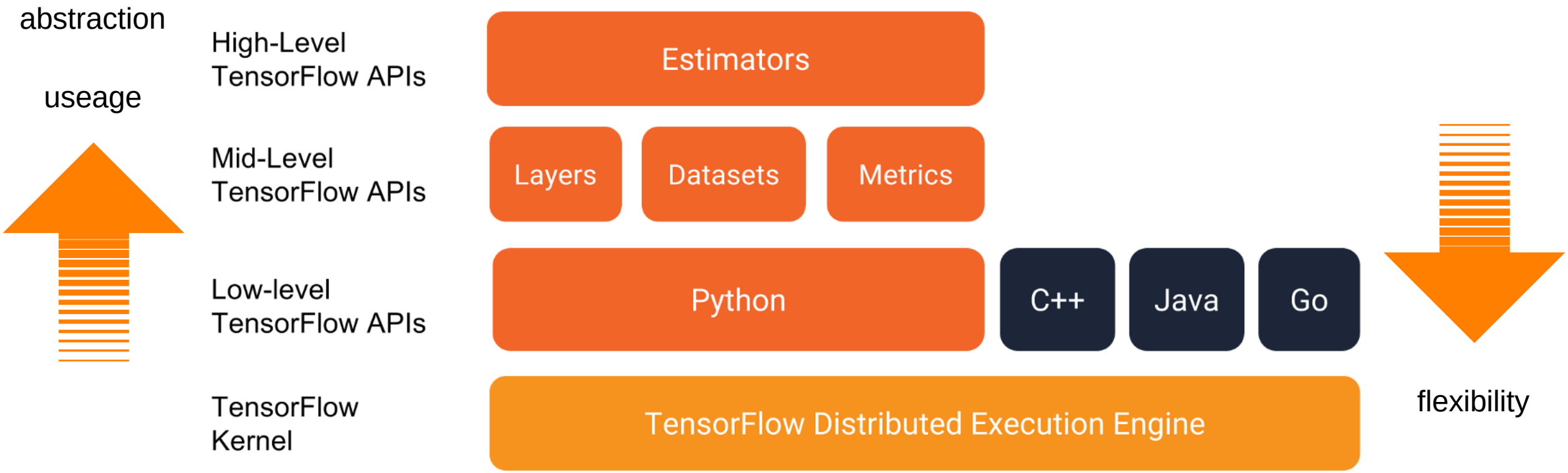
Saves / Load the model to Tensorflow SavedModel or a single HDF5 file.

```
models.save_model( model, ".\cnn_model" )


models.load_model(".\cnn_model")
```

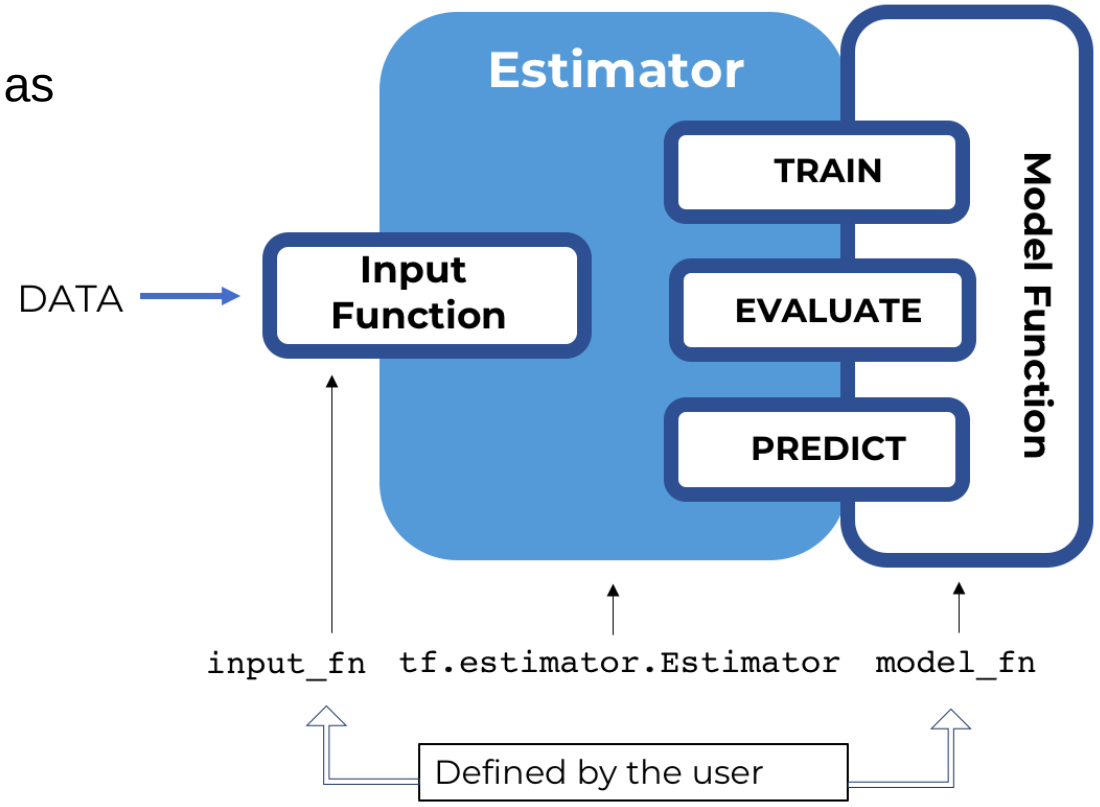
使用high-level TensorFlow API

programming stack of TF



A schematic of Estimator

Estimator hides some TensorFlow concepts, such as Graph and Session, from the user.



Estimators Interface

- **train-evaluate-predict** loop similar to scikit-learn
- The user using conditionals to denote behaviour that differs between TRAIN, EVALUATE and PREDICT upon which we can call .train, .eval, and .predict
- create a custom Estimator :
  - A model function model_fn that is fed the features, labels etc.
  - The model function defines model, loss, optimizer, and metrics.

mode parameter

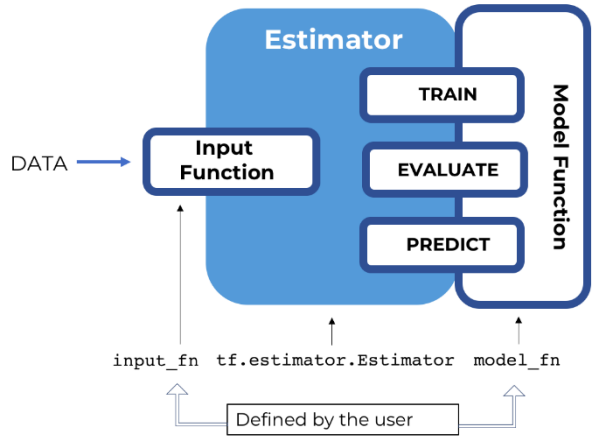
- The mode parameter can take one of three values:
  - tf.estimator.ModeKeys.TRAIN
  - tf.estimator.ModeKeys.EVAL
  - tf.estimator.ModeKeys.PREDICT

Training

```
# Calculate Loss (for both TRAIN and EVAL modes)
    loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)

    # Configure the Training Op (for TRAIN mode)
    if mode == tf.estimator.ModeKeys.TRAIN:
      optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
      train_op = optimizer.minimize(
        loss=loss,
        global_step=tf.train.get_global_step())
    return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)
```

Training with estimator

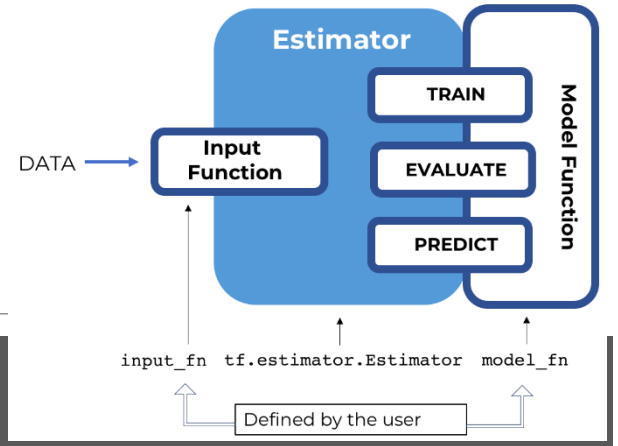



```
# Train the model
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": train_data},
    y=train_labels,
    batch_size=100,
    num_epochs=None,
    shuffle=True)
# train one step and display the probabilties
mnist_classifier.train(input_fn=train_input_fn, steps=1000)
```

Evaluation

```
# Add evaluation metrics (for EVAL mode)
    eval_metric_ops = {
      "accuracy": tf.metrics.accuracy(
          labels=labels, predictions=predictions["classes"])
    }
    return tf.estimator.EstimatorSpec(
      mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

Evaluation with estimator




```
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": eval_data},
    y=eval_labels,
    num_epochs=1,
    shuffle=False)

eval_results = mnist_classifier.evaluate(input_fn=eval_input_fn)
```

Monitoring progress during training

```
# Set up logging level
tf.logging.set_verbosity(tf.logging.INFO)


# 5 levels in order of increasing severity
# DEBUG
# INFO
# WARN
# ERROR
# FATAL
```

Code injection using Hooks

```
# Set up logging for predictions
tensors_to_log = {"probabilities": "softmax_tensor"}

logging_hook = tf.train.LoggingTensorHook(
    tensors=tensors_to_log, every_n_iter=50)
```

Train mode Estimator with hook

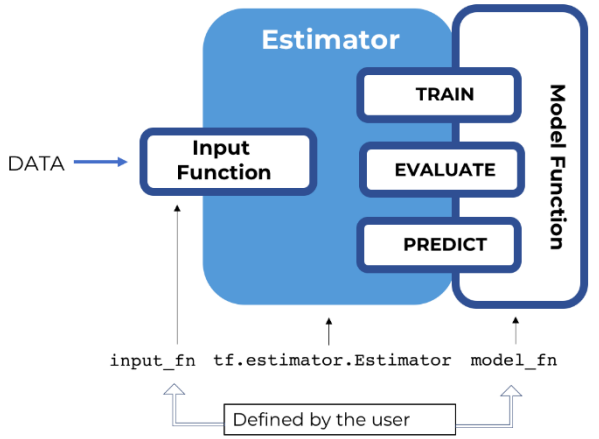
```
# train one step and display the probabilties
mnist_classifier.train(
    input_fn=train_input_fn,
    steps=1,
    hooks=[logging_hook])
```

Predictions

```
predictions = {
      # Generate predictions (for PREDICT and EVAL mode)
      "classes": tf.argmax(input=logits, axis=1),
      # Add `softmax_tensor` to the graph. It is used for PREDICT and by the
      # `logging_hook`.
      "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
    return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)
```

Predictions with estimator




```
pred_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": eval_data[0:10]},
    y=eval_labels[0:10],
    num_epochs=1,
    shuffle=False)
pred_results = mnist_classifier.predict(input_fn=pred_input_fn)
for p in enumerate(pred_results):
    print(p[0],p[1])
print(eval_labels[0:10])
```