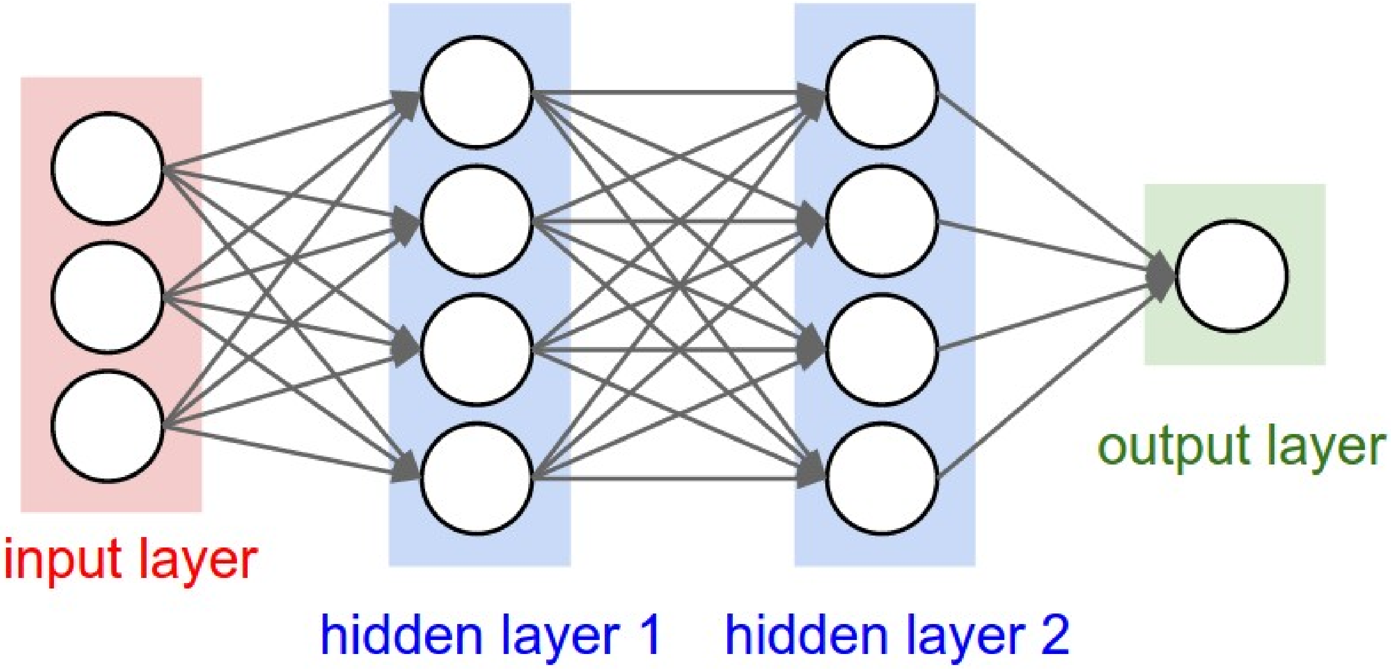
Neural Network Example

Build a 2-hidden layers fully connected neural network (a.k.a multilayer perceptron) with TensorFlow v2.

This example is using a low-level approach to better understand all mechanics behind building neural networks and the training process.

- Author: Aymeric Damien
- Project: https://github.com/aymericdamien/TensorFlow-Examples/

Neural Network Overview



MNIST Dataset Overview

This example is using MNIST handwritten digits. The dataset contains 60,000 examples for training and 10,000 examples for testing. The digits have been size-normalized and centered in a fixed-size image (28x28 pixels) with values from 0 to 255.

In this example, each image will be converted to float32, normalized to [0, 1] and flattened to a 1-D array of 784 features (28*28).

MNIST Dataset

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

```
from __future__ import absolute_import, division, print_function
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.keras import Model, layers
import numpy as np
import pandas as pd        # For loading and processing the dataset.
from sklearn.model_selection import train_test_split
```

```
# MNIST dataset parameters.
num_classes = 2 # total classes (0-9 digits).
num_features = 14 # data features (img shape: 28*28).

# Training parameters.
learning_rate = 0.1
training_steps = 2000
batch_size = 256
display_step = 100

# Network parameters.
n_hidden_1 = 128 # 1st layer number of neurons.
n_hidden_2 = 256 # 2nd layer number of neurons.
```

```
# Finally, we convert the Pandas dataframe to a NumPy array, and split it into a
# Read the CSV input file and show first 5 rows
df_train = pd.read_csv('/content/task1_dataset_train.csv')
df_train.head(5)
X = df_train.drop('y', axis=1).values
y = df_train['y'].values
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scale, y, test_size=0.2)
```

```
# Use tf.data API to shuffle and batch data.
train_data = tf.data.Dataset.from_tensor_slices((X_train, y_train))
train_data = train_data.repeat().shuffle(5000).batch(batch_size).prefetch(1)
```

```
# Create TF Model.
class NeuralNet(Model):
    # Set layers.
    def __init__(self):
        super(NeuralNet, self).__init__()
        # First fully-connected hidden layer.
        self.fc1 = layers.Dense(n_hidden_1, activation=tf.nn.relu)
        # First fully-connected hidden layer.
        self.fc2 = layers.Dense(n_hidden_2, activation=tf.nn.relu)
        # Second fully-connecter hidden layer.
        self.out = layers.Dense(num_classes)

    # Set forward pass.
    def call(self, x, is_training=False):
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.out(x)
        if not is_training:
            # tf cross entropy expect logits without softmax, so only
            # apply softmax when not training.
            x = tf.nn.softmax(x)
        return x

# Build neural network model.
neural_net = NeuralNet()
```

```
# Cross-Entropy Loss.
# Note that this will apply 'softmax' to the logits.
def cross_entropy_loss(x, y):
    # Convert labels to int 64 for tf cross-entropy function.
    y = tf.cast(y, tf.int64)
    # Apply softmax to logits and compute cross-entropy.
    loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y, logits=x)
    # Average loss across the batch.
    return tf.reduce_mean(loss)

# Accuracy metric.
def accuracy(y_pred, y_true):
    # Predicted class is the index of highest score in prediction vector (i.e. arg
    correct_prediction = tf.equal(tf.argmax(y_pred, 1), tf.cast(y_true, tf.int64))
    return tf.reduce_mean(tf.cast(correct_prediction, tf.float32), axis=-1)

# Stochastic gradient descent optimizer.
optimizer = tf.optimizers.SGD(learning_rate)
```

```
# Optimization process.
def run_optimization(x, y):
    # Wrap computation inside a GradientTape for automatic differentiation.
    with tf.GradientTape() as g:
        # Forward pass.
        pred = neural_net(x, is_training=True)
        # Compute loss.
        loss = cross_entropy_loss(pred, y)

    # Variables to update, i.e. trainable variables.
    trainable_variables = neural_net.trainable_variables

    # Compute gradients.
    gradients = g.gradient(loss, trainable_variables)

    # Update W and b following gradients.
    optimizer.apply_gradients(zip(gradients, trainable_variables))
```

```
# Run training for the given number of steps.
for step, (batch_x, batch_y) in enumerate(train_data.take(training_steps), 1):
    # Run the optimization to update W and b values.
    run_optimization(batch_x, batch_y)

    if step % display_step == 0:
        pred = neural_net(batch_x, is_training=True)
        loss = cross_entropy_loss(pred, batch_y)
        acc = accuracy(pred, batch_y)
        print("step: %i, loss: %f, accuracy: %f" % (step, loss, acc))
```

```
step: 100, loss: 0.486165, accuracy: 0.769531
step: 200, loss: 0.520279, accuracy: 0.750000
step: 300, loss: 0.530207, accuracy: 0.714844
step: 400, loss: 0.465484, accuracy: 0.789062
step: 500, loss: 0.461355, accuracy: 0.792969
step: 600, loss: 0.475713, accuracy: 0.789062
step: 700, loss: 0.436438, accuracy: 0.804688
step: 800, loss: 0.450595, accuracy: 0.757812
step: 900, loss: 0.489487, accuracy: 0.765625
step: 1000, loss: 0.461266, accuracy: 0.750000
step: 1100, loss: 0.420362, accuracy: 0.804688
step: 1200, loss: 0.434839, accuracy: 0.812500
step: 1300, loss: 0.410994, accuracy: 0.781250
step: 1400, loss: 0.398383, accuracy: 0.800781
step: 1500, loss: 0.347369, accuracy: 0.851562
step: 1600, loss: 0.385559, accuracy: 0.796875
step: 1700, loss: 0.425821, accuracy: 0.789062
step: 1800, loss: 0.377654, accuracy: 0.824219
step: 1900, loss: 0.424663, accuracy: 0.769531
step: 2000, loss: 0.415788, accuracy: 0.796875
```

```
# Test model on validation set.
pred = neural_net(X_test, is_training=False)
print("Test Accuracy: %f" % accuracy(pred, y_test))
```

```
Test Accuracy: 0.807301
```

```
df_test = pd.read_csv('/content/task1_dataset_test.csv')
df_test.head()
```

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_10	x_11	x_12	x_13	x_14
0	57	2	21598	1	1	6	0	0	0	1	1	0	19	0
1	27	2	969	0	0	0	9	0	0	1	1	0	19	0
2	25	6	1126	1	1	1	6	1	0	0	12	0	4	0

```
df_test = df_test.values
min_max_scaler = preprocessing.MinMaxScaler()
df_test_scale = min_max_scaler.fit_transform(df_test)
```

```
X_test
```

```
array([[0.27777778, 0.25      , 0.15548456, ..., 0.        , 0.        ,
        0.        ],
       [0.125     , 0.25      , 0.12631384, ..., 0.03296703, 0.04301075,
        0.        ],
       [0.79166667, 0.625     , 0.11913692, ..., 0.        , 0.29032258,
        0.        ],
       ...,
       [0.55555556, 0.625     , 0.61994721, ..., 0.        , 0.        ,
        0.        ],
       [0.06944444, 0.125     , 0.20924202, ..., 0.        , 0.04301075,
        0.        ],
       [0.40277778, 0.25      , 0.69912488, ..., 0.        , 0.        ,
        0.        ]])
```

```
# Predict 5 images from validation set.
n_t = 100
test_t = df_test_scale[:n_t]
predictions = neural_net(test_t)

# Display image and model prediction.
for i in range(n_t):
    print("Model prediction: %i" % np.argmax(predictions.numpy()[i]))
```

```
Model prediction: 0
Model prediction: 0
Model prediction: 1
Model prediction: 0
Model prediction: 1
Model prediction: 0
Model prediction: 1
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
Model prediction: 0
```

```
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 1
    Model prediction: 0
    Model prediction: 1
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 1
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 1
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 0
    Model prediction: 1
    Model prediction: 0
    Model prediction: 0
```

Author: Aymeric Damien

✓ 0s completed at 12:09 PM ● ×