LAMPIRAN

Lampiran 1 CNN6_MNIST.py

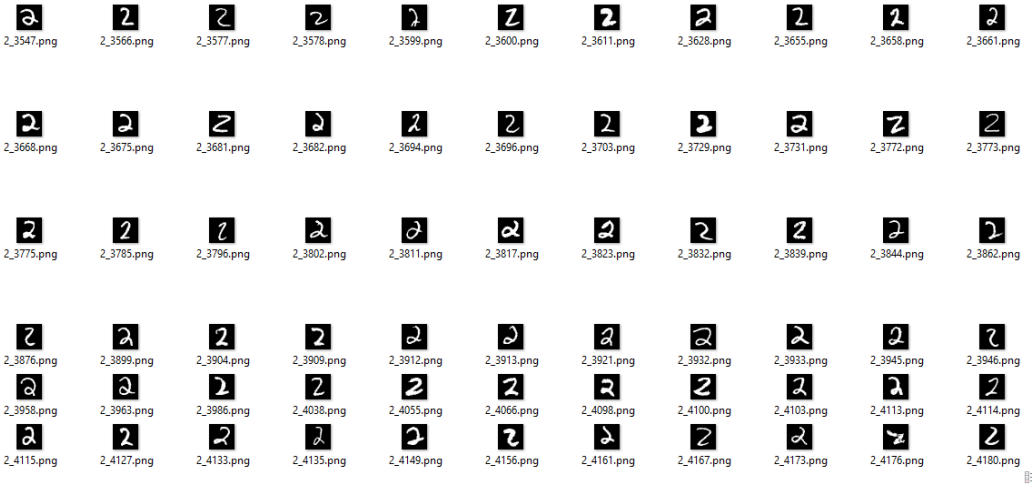
CNN6_MNIST.py

```
import os
import random
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

class CNN6_MNIST:
    def read_MNIST_dataset(self):
        self.mnist = input_data.read_data_sets('MNIST_data',
one_hot=True)

    def predict(self, image):
        return tf.cast(tf.argmax(self.prediction, 1),
tf.float32).eval(session=self.session, feed_dict={self.x:
[image]})[0]

    def predict_random_from_MNIST(self):
        try: self.mnist
        except AttributeError: self.read_MNIST_dataset()
        image = self.mnist.test.images[random.randint(0,
9999)]
        return (image, self.predict(image))

    def build(self, train=False,
checkpoint_dir='./checkpoints/', training_steps=1000):
        self.x = tf.placeholder("float", shape=[None, 784])
        rgb = tf.reshape(self.x, [-1, 28, 28, 1])

        conv_1 = self.conv_layer(rgb, 'conv_1')
        pool_1 = self.pool_layer(conv_1)

        conv_2 = self.conv_layer(pool_1, 'conv_2')
        pool_2 = self.pool_layer(conv_2)

        fc_1 = self.fc_layer(tf.reshape(pool_2, [-1, 7 * 7 *
64]), 'fc_1')

        if train:
            fc_1 = tf.nn.dropout(fc_1, 0.5)

        self.prediction = self.fc_layer(fc_1, 'fc_2')
        y_ = tf.placeholder("float", shape=[None, 10])

        self.cross_entropy = tf.reduce_mean(-
tf.reduce_sum(y_ * tf.log(self.prediction),
reduction_indices=[1]))
```

```
        self.train_step = tf.train.AdamOptimizer(1e-
4).minimize(self.cross_entropy)
        self.correct_prediction =
tf.equal(tf.argmax(self.prediction, 1), tf.argmax(y_, 1))
        self.accuracy =
tf.reduce_mean(tf.cast(self.correct_prediction, tf.float32))

        saver = tf.train.Saver(max_to_keep=5,
keep_checkpoint_every_n_hours=1)
        self.session = tf.Session()

        if train:
            try: self.mnist
            except AttributeError: self.read_MNIST_dataset()

            self.session.run(tf.initialize_all_variables())
            for i in range(training_steps):
                batch = self.mnist.train.next_batch(50)
                self.train_step.run(session=self.session,
feed_dict={self.x: batch[0], y_: batch[1]})
                if i%100 == 0:
                    train_accuracy =
self.accuracy.eval(session=self.session, feed_dict={self.x:
batch[0], y_: batch[1]})
                    print("step {}, training accuracy
{}".format(i, train_accuracy))
                    saver.save(self.session,
checkpoint_dir+'model', global_step=i)

            print("test accuracy
{}".format(self.accuracy.eval(session=self.session,
feed_dict={self.x: self.mnist.test.images, y_:
self.mnist.test.labels})))
            print("Model saved in file: ", checkpoint_dir)
        else:
            self.session.run(tf.initialize_all_variables())

            if not os.path.exists(checkpoint_dir):
                raise IOError(checkpoint_dir+' does not
exist.')
            else:
                path =
tf.train.get_checkpoint_state(checkpoint_dir)
                if path is None:
                    raise IOError('No checkpoint to restore
in '+checkpoint_dir)
                else:
                    saver.restore(self.session,
path.model_checkpoint_path)
```

```
    def weight_variable(self, shape):
      initial = tf.truncated_normal(shape, stddev=0.1)
      return tf.Variable(initial)

    def bias_variable(self, shape):
      initial = tf.constant(0.1, shape=shape)
      return tf.Variable(initial)

    def conv2d(self, x, W):
      return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1],
padding='SAME')

    def conv_layer(self, x, name):
        if name == 'conv_1':
            W = self.weight_variable([5, 5, 1, 32])
            b = self.bias_variable([32])
        elif name == 'conv_2':
            W = self.weight_variable([5, 5, 32, 64])
            b = self.bias_variable([64])
        else:
            raise ValueError(name+' is not part of the
model')
        return tf.nn.relu(self.conv2d(x, W) + b)

    def pool_layer(self, x):
        return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
strides=[1, 2, 2, 1], padding='SAME')

    def fc_layer(self, x, name):
        if name == 'fc_1':
            W = self.weight_variable([7 * 7 * 64, 1024])
            b = self.bias_variable([1024])
            return tf.nn.relu(tf.matmul(x, W) + b)
        elif name == 'fc_2':
            W = self.weight_variable([1024, 10])
            b = self.bias_variable([10])
            return tf.nn.softmax(tf.matmul(x, W) + b)
        else:
            raise ValueError(name+' is not part of the
model')
```

Lampiran 2 Data Uji



G

ambar lampiran Data Uji.