def get\_model():

base\_model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(filters=16, kernel\_size=(5,5), strides=(2,2), activation='relu', input\_shape=(224,224,3)),

tf.keras.layers.BatchNormalization(),

#tf.keras.layers.MaxPool2D(pool\_size=(3,3), strides=(2,2)),

tf.keras.layers.AveragePooling2D(pool\_size=(3,3), strides=(2,2)),

tf.keras.layers.Conv2D(filters=32, kernel\_size=(5,5), strides=(1,1), activation='relu', padding="same"),

tf.keras.layers.BatchNormalization(),

#tf.keras.layers.MaxPool2D(pool\_size=(3,3), strides=(2,2)),

tf.keras.layers.AveragePooling2D(pool\_size=(3,3), strides=(2,2)),

#tf.keras.layers.Conv2D(filters=64, kernel\_size=(3,3), strides=(1,1), activation='relu', padding="same"),

#tf.keras.layers.BatchNormalization(),

#tf.keras.layers.Conv2D(filters=128, kernel\_size=(3,3), strides=(1,1), activation='relu', padding="same"),

#tf.keras.layers.BatchNormalization(),

#tf.keras.layers.Conv2D(filters=80, kernel\_size=(3,3), strides=(1,1), activation='relu', padding="same"),

#tf.keras.layers.BatchNormalization(),

#tf.keras.layers.MaxPool2D(pool\_size=(3,3), strides=(2,2)),

tf.keras.layers.Flatten(),

#tf.keras.layers.Dense(4, activation='relu'),

#tf.keras.layers.Dropout(0.5),

#tf.keras.layers.Dense(2, activation='relu'),

#tf.keras.layers.Dropout(0.5),

tf.keras.layers.Dense(64, activation='relu'),

])

out = base\_model.output

prediction = tf.keras.layers.Dense(1, activation="linear")(out)

model = tf.keras.Model(inputs = base\_model.input, outputs = prediction)

return model

def get\_model\_name(k):

return 'NewNet\_'+str(k)+'.h5'

tf.test.gpu\_device\_name()