|  |  |
| --- | --- |
| **model = tf.keras.Sequential( [**  **tf.keras.layers.Input(shape=** **shape=(28, 28)),**  **tf.keras.layers.Flatten(),**  **tf.keras.layers.Normalization()**  **tf.keras.layers.Dense(300, activation="relu", kernel\_initializer="he\_normal", kernel\_regularizer=tf.keras.regularizers.l2(0.01)), use\_bias=False),**  **tf.keras.layers.BatchNormalization(),**  **tf.keras.layers.Activation("relu"),**  **tf.keras.layers.Concatenate()**  **tf.keras.layers.Dropout(rate=0.2),**  **])** | **capa1= Dense(300, activation="relu")**  **model.add(capa1)** |
| **model.add(Dense(300, activation="relu"))** |

**optimizer = tf.keras.optimizers.Nadam(learning\_rate=1e-5)**

**model.compile(loss="sparse\_categorical\_crossentropy",optimizer=optimizer, metrics=["accuracy"])**

**checkpoint\_cb = tf.keras.callbacks.ModelCheckpoint("Resultados/my\_checkpoints.weights.h5", save\_weights\_only=True)**

**early\_stopping\_cb = tf.keras.callbacks.EarlyStopping(patience=10, restore\_best\_weights=True**

**history = model.fit(X\_train, y\_train, epochs=30, validation\_data=(X\_valid, y\_valid), callbacks = [early\_stopping\_cb, model\_checkpoint\_cb])**

**model.evaluate(X\_test, y\_test)**

**y\_pred = model.predict(X\_new)**