Deep Learning

ex. 1

Haim Adrian,
Lecturer: Dr. Igor Nor

val_accuracy: 0.9307
trainable params: 1,726,474

Hyperparameters Tuning

לאחר מספר ניסיונות ידניים של מבנה הרשת ופרמטרים, הוגדרו מספר ערכים אפשריים אשר הועברו ל-GridSearchCV על מנת למצוא את הצירוף האופטימאלי מבין הפרמטרים שנבחרו.
מפאת חוסר הזמן, הוגדרו סטים קבועים של ערכים אפשריים. הייתי מוסיף יותר ערכים ע"י שימוש ב- range, או לולאות, כדי למצוא ערכים שלא חשבתי עליהם לצורך הבחירה בסט האופטימאלי.

בחירת Optimizer - Adam, Adadelta, Adagrad, RMSprop, SGD
פרמטרים
1.0, 0.1, 0.01, 0.001, 0.0001 – learning_rate
0.99, 0.9, 0.8 – momentum

בחירת מבנה הרשת
שכבה נסתרת אחת עם 1568 נוירונים (2*28*28)
6 שכבות נסתרות, עם כמות נוירונים בסדר הבא: (128, 256, 512, 512, 512, 512)
6 שכבות נסתרות, עם כמות נוירונים בסדר הבא: (512, 512, 512, 512, 512, 512)
17 שכבות נסתרות: (256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256)
היערה: השכבה האחת קיבלה ציון מצוין באופן קבוע, ונלמדה באופן מהיר הרבה יותר, למרות שאמרנו שעדיף יותר שכבות על פני יותר נוירונים בשכבה. מניח שניתן למצוא רשת עמוקה יותר שלא ניסתי.
בנוסף נעשתה בדיקה עם אקטיבציה לפני נירמול, ואקטיבציה כשכבה נפרדת, לאחר נירמול.

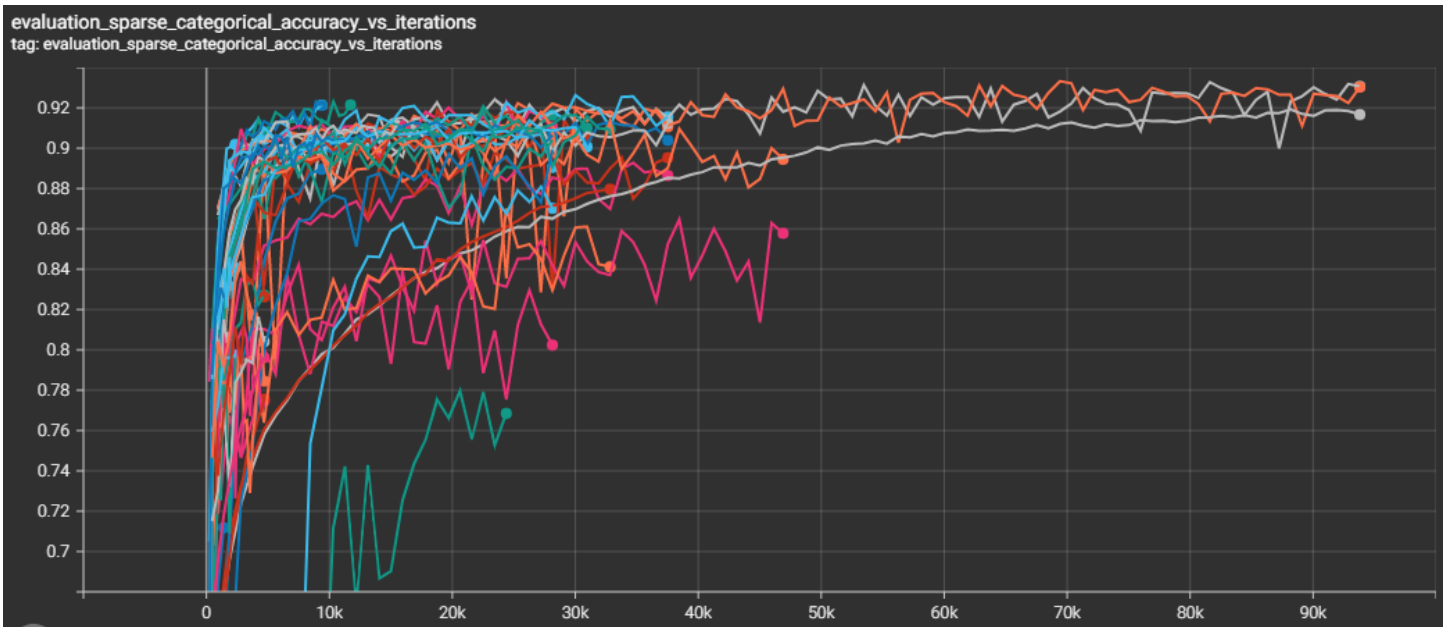
גודל Batch
64, 128, 256

כמות epoch-ים
20, 30, 50, 100
כמות גדולה של epoch-ים שיפרה וייצבה את Adagrad, שלקח לו פי 3 יותר epoch-ים כדי להתייצב מעל 90%.

אתחול משקולות, אקטיבציה, רגולריזציה והמנעות מ-overfitting
he_uniform, he_normal, glorot_uniform – kernel_initializer
0.01, 0.001, 0.0001 – kernel_regularizer
0, 0.3, 0.4 – dropout
relu, selu, tanh – activation

Loss - categorical_crossentropy
שיערוך – accuracy

דוגמאות הרצה



שתי רשתות נבחרו: (מאמנות 1,726,474 פרמטרים)
Adagrad עם ציון של 93%, עם 75-100 epoch-ים.
Adadelta עם ציון של 92% תוך 25 epoch-ים.

Adagrad

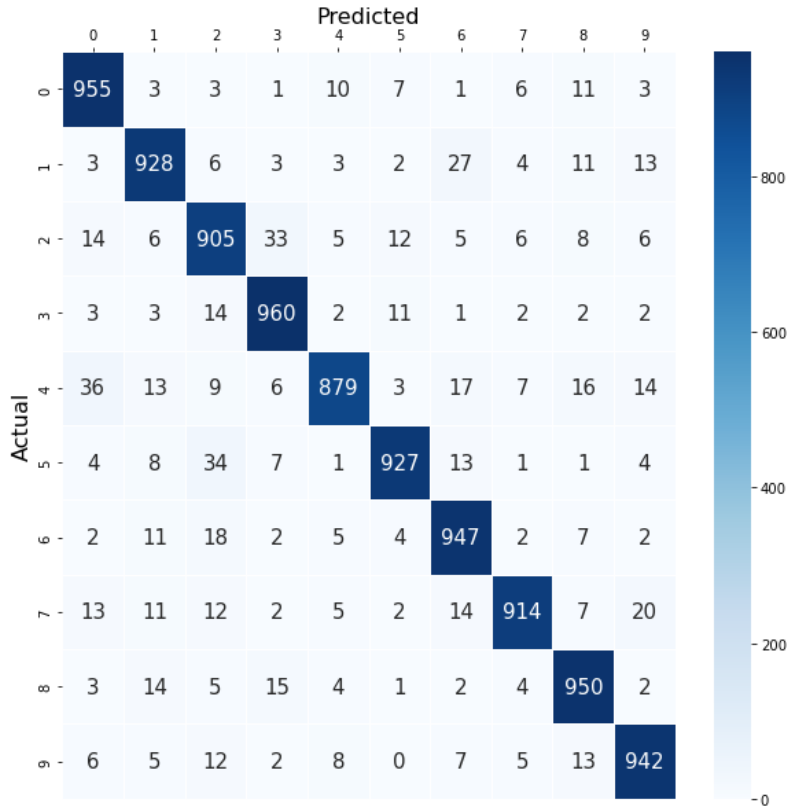
רשת: 784 -> (512 -> 512 -> 512 -> 512 -> 512 -> 512) -> 10
Batch: 64
epoch: 75
learning_rate: 0.01
kernel_initializer: glorot_uniform
kernel_regularizer: 0.001
dropout: 0 (ללא)
activation: relu
BatchNormalization לפני אקטיבציה

Adadelta

רשת: 784 -> (512 -> 512 -> 512 -> 512 -> 512 -> 512) -> 10
Batch: 128
epoch: 25
learning_rate: 1.0
kernel_initializer: he_normal
kernel_regularizer: 0.0001
dropout: 0.3
activation: relu
BatchNormalization לפני אקטיבציה

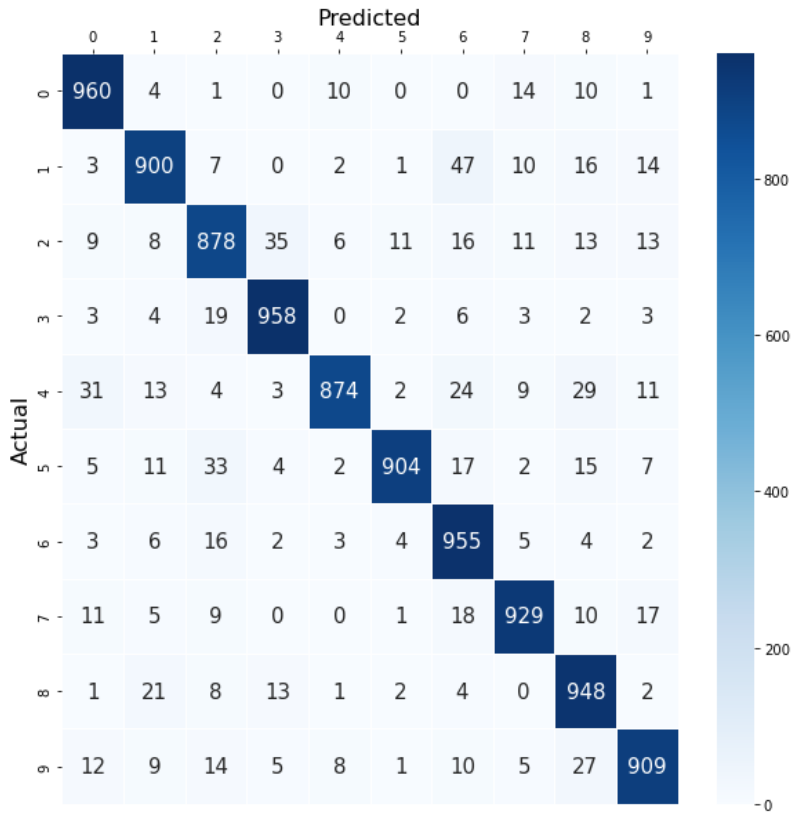
Adagrad – 93%

Confusion Matrix on KMNIST predictions



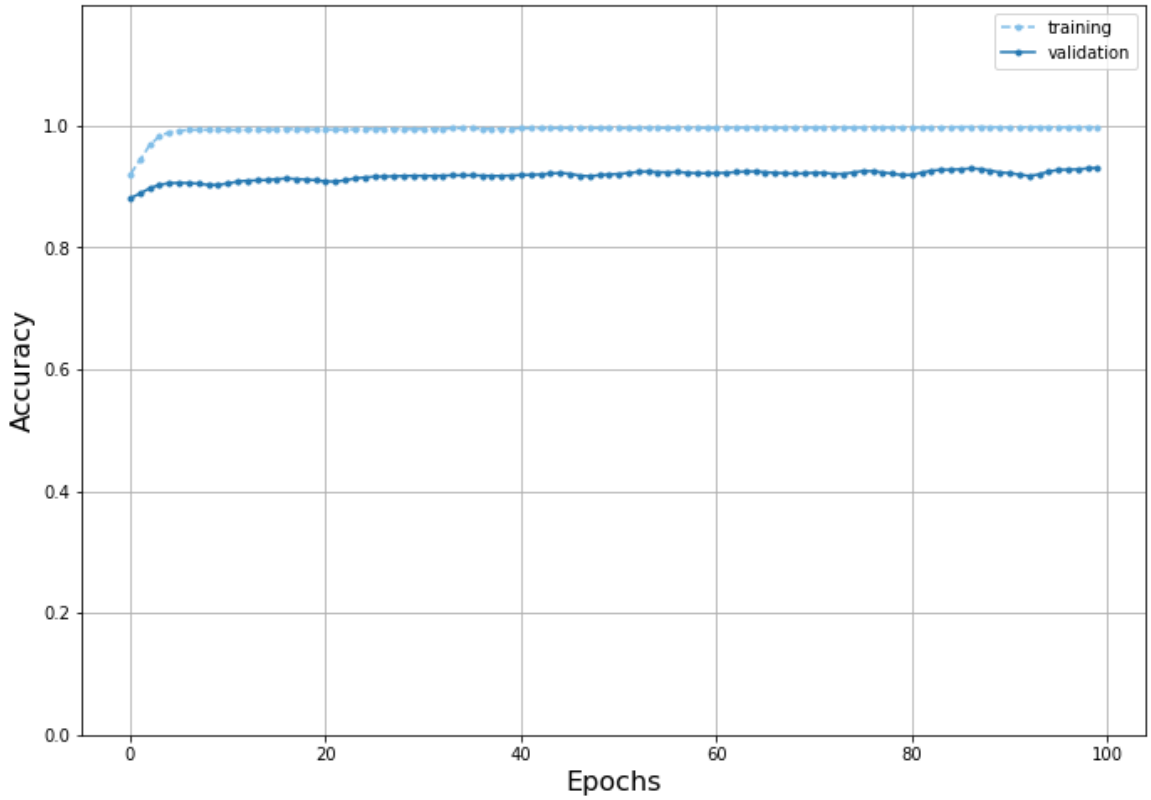
Adadelta – 92%

Confusion Matrix on KMNIST predictions

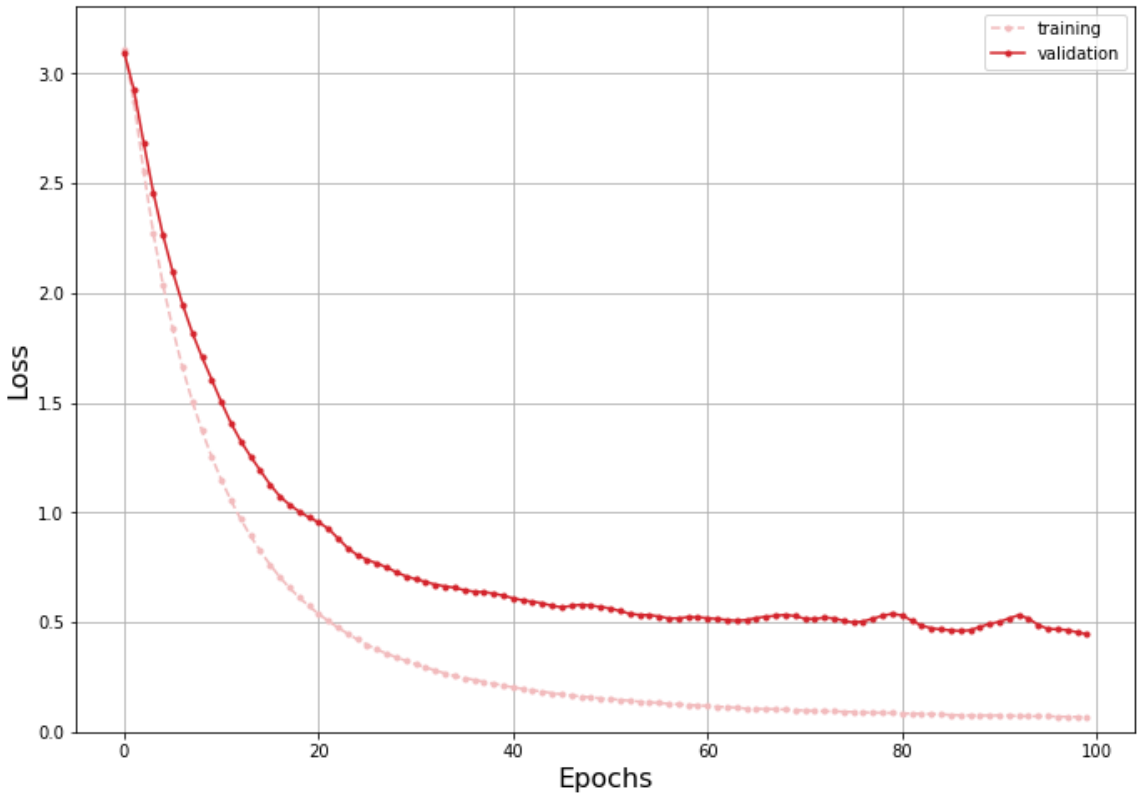


Adagrad

Accuracy: 0.9307

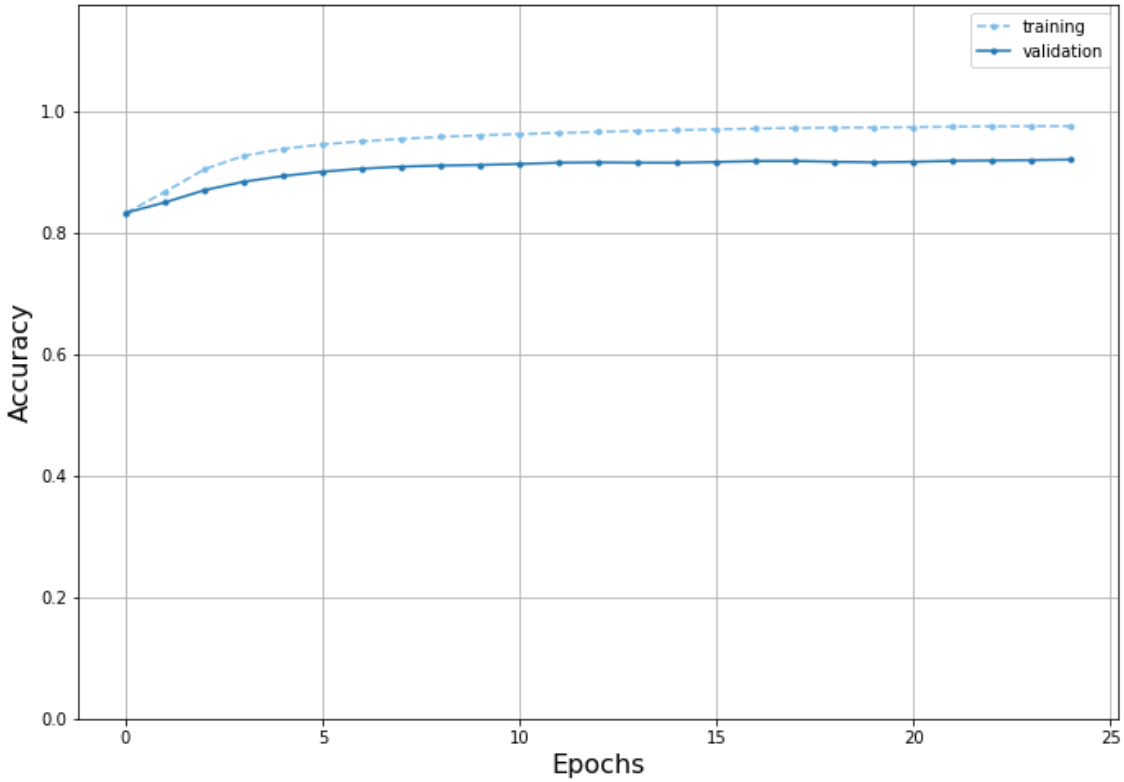


Loss: 0.4424

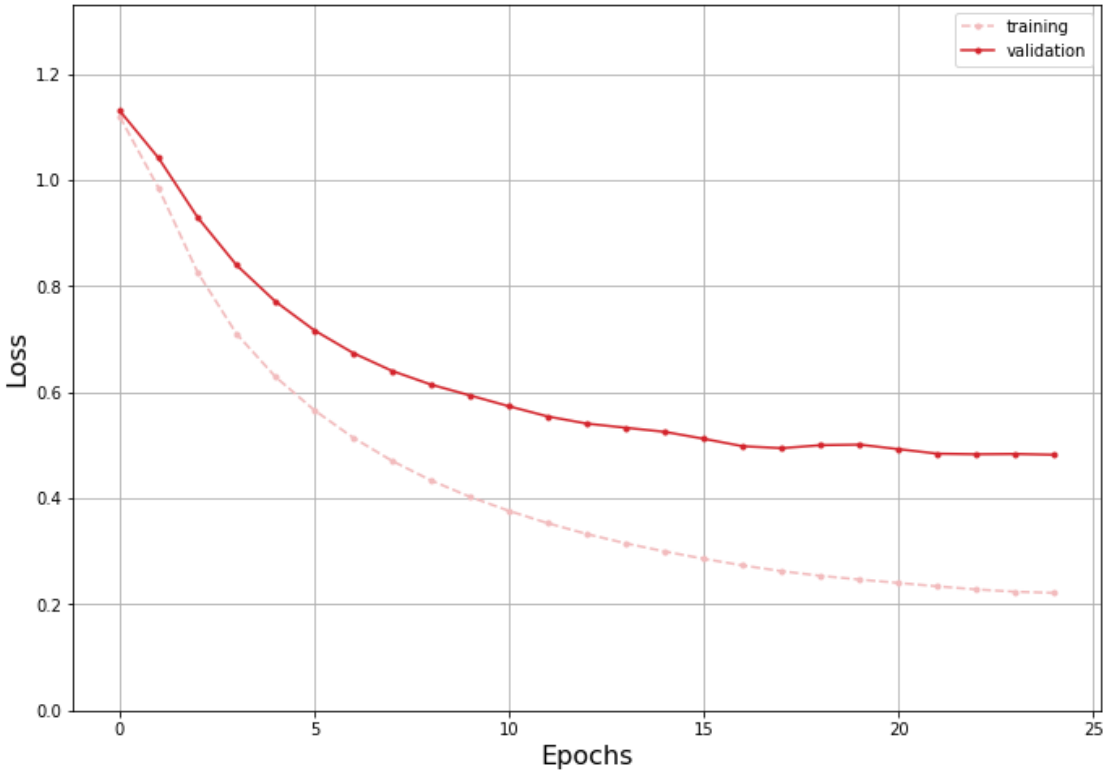


Adadelta

Accuracy: 0.9215



Loss: 0.4777



נספח - הקוד עבור בחירת פרמטרים

```
import numpy as np
import tensorflow_datasets as tfds
from tensorflow.keras import regularizers
from tensorflow.keras.layers import Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adadelta, Adam, RMSprop, Adagrad
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV

# ------------------ Load Data ---------------------------------------
# Read all data (batch_size=-1), to convert it to np array
(ds_train, ds_test), ds_info = tfds.load('kmnist', split=['train', 'test'], shuffle_files=True, as_supervised=True,
with_info=True, batch_size=-1)
(X_train, y_train) = tfds.as_numpy(ds_train)
(X_test, y_test) = tfds.as_numpy(ds_test)

# Reshape from (60000, 28, 28, 1) to (60000, 784) - This is the flattening of images
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = np.float64(X_train) / 255.
X_test = np.float64(X_test) / 255.
# ---------------------------------------------------------------------
def create_model(layers, activation, kernel_initializer, kernel_regularizer, dropout, optimizer):
    opt = None
    if optimizer[0] == 'Adadelta':
        opt = Adadelta(learning_rate=optimizer[1])
    elif optimizer[0] == 'RMSprop':
        opt = RMSprop(learning_rate=optimizer[1])
    elif optimizer[0] == 'Adagrad':
        opt = Adagrad(learning_rate=optimizer[1])
    else:
        opt = Adam(learning_rate=optimizer[1])
    model = Sequential()
    for i, units in enumerate(layers):
        if i==0:
            model.add(Dense(units=units, kernel_initializer=kernel_initializer,
kernel_regularizer=regularizers.l2(kernel_regularizer), input_dim=X_train.shape[1]))
        else:
            model.add(Dense(units=units, kernel_initializer=kernel_initializer,
kernel_regularizer=regularizers.l2(kernel_regularizer)))
        model.add(BatchNormalization())
        model.add(Dropout(dropout))
        model.add(Activation(activation))

    model.add(Dense(10, activation='softmax'))
    model.compile(optimizer=opt, loss=SparseCategoricalCrossentropy(from_logits=False), metrics=['accuracy'])
    model.summary()
    return model
```

```python
# Keras classifier that applies our create_model method
classifier = KerasClassifier(build_fn=create_model, verbose=0)

# Tuning some parameters
param_grid = dict(layers=[(784 * 2,), (128, 256, 512, 512, 512, 512), (512, 512, 512, 512, 512, 512), (256, 256, 256, 256, 256,
256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256)],
                  activation=['relu', 'selu', 'tanh'],
                  kernel_initializer=['he_uniform', 'he_normal', 'glorot_uniform'],
                  kernel_regularizer=[1e-2, 1e-3, 1e-4],
                  dropout=[0, 0.3, 0.4],
                  optimizer=[('Adam', 1e-4), ('Adam', 1e-3), ('Adam', 1e-2), ('Adadelta', 1.0), ('RMSprop', 1e-3), ('Adagrad',
1e-4), ('Adagrad', 1e-3), ('Adagrad', 1e-2)],
                  batch_size=[64, 128, 256],
                  epochs=[20, 30, 50, 100])

# Using GridSearchCV to fit the param dictionary
grid_search = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=3, scoring='accuracy', n_jobs=24, verbose=3)
grid_search = grid_search.fit(X_train, y_train)
print(grid_search)
print(f'Best params={grid_search.best_params_}. \nBest score: {grid_search.best_score_}')
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
params = grid_search.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

פלט GridSearch:

```
Best params={'activation': 'relu', 'batch_size': 64, 'dropout': 0, 'epochs': 100, 'kernel_initializer': 'glorot_uniform',
'kernel_regularizer': 0.01, 'layers': (512, 512, 512, 512, 512, 512), 'optimizer': ('Adagrad', 0.01)}.
Best score: 0.9677

Best params={'activation': 'relu', 'batch_size': 128, 'dropout': 0.3, 'epochs': 20, 'kernel_initializer': 'he_normal',
'kernel_regularizer': 0.0001, 'layers': (512, 512, 512, 512, 512, 512), 'optimizer': ('Adadelta', 1.0)}.
Best score: 0.9636
```