Fashion Image Recognition Project

DECEMBER 14TH, 2021

Syracuse University

IST 707

Noah Laraway

Ross Conway

Anya Patel

Table of Contents

Report Overview

Machine learning with visual AI has recently gained tremendous growth in a wide variety of industries. From analyzing medical images in hospitals to detecting quality issues in manufacturing plants the applications for machine vision are endless. One industry in particular where machine vision is used to identify objects is in the fashion industry. Since e-commerce has become so important to the retail industry, application use cases like searching and tagging clothing and deciding the correct measurement of shoes and attire has led to the adoption of image recognition. This project takes photo data for clothing and uses machine learning and image recognition to predict the class label for that image, e.g., dress, sneaker or coat.

Data Acquisition, Loading and Cleansing

The data for this project was downloaded from the Fashion MNIST dataset on Kaggle: Kaggle-Fashion-MNIST. There are 2 CSV files in the dataset with one being for the train data and the other for the test data. Both CSV files contains 785 columns. The train set contains 60,000 rows and the test set contains 10,000 rows where each row is a separate image. The first column is the class label which contains a number, from 0 to 9, that represents the article of clothing in the image. The remaining columns are the pixels in the image. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The data was imported via mnist_reader: fashion-mnist/mnist_reader.py at master · zalandoresearch/fashion-mnist · GitHub. From there preprocessing was a relatively simple task and we were able to begin exploring our data set and experimenting with our models.

Initial Analysis

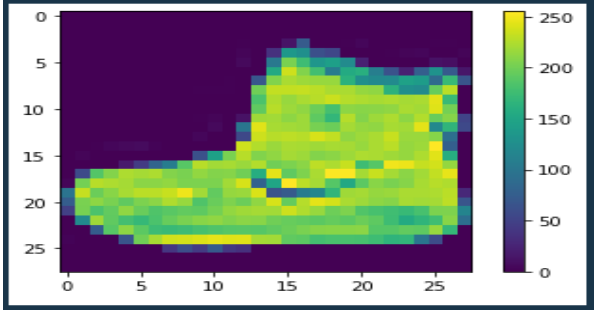
Initial visualization was performed to explore the data. Some images of the exploration are below.

Plot showing various images and labels:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

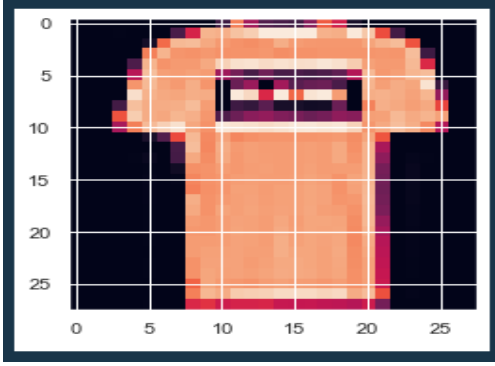
Grid plot of a sneaker

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



Grid plot of a T-shirt/top

```
plt.imshow(x_train[1].reshape((28,28)))
plt.show()
```



Modeling and Results

Models used (9 Total): Random Forest, K-Nearest Neighbors, Linear Model, Support Vector Model, Logistic Regression, Gaussian Naïve Bayes, Decision Tree, Keras Tensor Flow, XGBoost

Random Forest - Number of Models Ran: 3, Best Score: 0.88, Time: 66.95s

```
# http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf
# MODEL BUILD - RANDOM FORESTS
start = time.time()
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 1000)
forest.fit(X_train, y_train)
print('Score: ', forest.score(X_test, y_test))
predictions = forest.predict(X_test)
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Score:  0.8793
Training time: 663.41490077972415s
```

K-Nearest Neighbors - Number of Models Ran: 3, Best Score: 0.85, Time: 1250.92s

```
# !!!Dont run either!!!
# ok run it but its slow
# with just a sample
start = time.time()
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train)
neigh.score(X_test, y_test)
print('Score: ', neigh.score(X_test, y_test))
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Score:  0.8541
Training time: 1250.9213700294495s
```

Linear Model - Number of Models Ran: 186, Best Score: 0.83, Time: 109.85s

```
# MODEL BUILD - LINEAR

from sklearn import linear_model
start = time.time()
lmf = linear_model.SGDClassifier(max_iter=1000)
lmf.fit(X_train, y_train)
lmf.score(X_test, y_test)
print('Score: ', lmf.score(X_test, y_test))
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Score:  0.7958
Training time: 109.84839272499084s
```

Support Vector Model - Number of Models Ran: 1, Best Score: 0.88, Time: 812.27s

```
# MODEL BUILD - SUPPORT VECTOR
from sklearn.svm import SVC
start = time.time()
svm = SVC()
svm.fit(X_train, y_train)
svm.score(X_test, y_test)
print('Score: ', svm.score(X_test, y_test))
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Score:  0.8828
Training time: 812.2729923725128s
```

Logistic Regression - Number of Models Ran: 1, Best Score: 0.84, Time: 147.96s

```
# log regression on new sample set
from sklearn.linear_model import LogisticRegression
start = time.time()
logReg = LogisticRegression(max_iter=2000)
logReg.fit(x_train,y_train)
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Training time: 147.9576280117035s
```

```
prediction_test  = logReg.predict(x_test)
prediction_train = logReg.predict(x_train)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precision_score, recall_score
from sklearn.metrics import classification_report
```

```
print(accuracy_score(y_train, prediction_train))
```

```
0.8810333333333333
```

```
print(accuracy_score(y_test, prediction_test))
```

```
0.8441
```

Gaussian Naïve Bayes - Number of Models Ran: 1, Best Score: 0.59, Time: 0.54s

```
# MODEL
# Initialize our classifier
gnb = GaussianNB()
# Train our classifier
start = time.time()
model = gnb.fit(X_train, y_train)
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Training time: 0.5406081676483154s
```

```
# MODEL
# Make predictions
start = time.time()
preds = gnb.predict(X_test)
stop = time.time()
print(f"Training time: {stop - start}s")
# print(preds)
```

```
Training time: 0.7485842704772949s
```

```
# MODEL
# Evaluate accuracy
print(accuracy_score(y_test, preds))
```

```
0.5856
```

Decision Tree - Number of Models Ran: 7, Best Score: 0.81, Time: 27s

```
# MODEL BUILD - DECISION TREE - Using Entropy (Adjust Max Depth)
start = time.time()
from sklearn.tree import DecisionTreeClassifier
decisionTree = DecisionTreeClassifier(criterion = "entropy",
                                      max_depth=12)

decisionTree.fit(X_train, y_train)
print('Score: ', decisionTree.score(X_test, y_test))
predictions = decisionTree.predict(X_test)
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
Score:  0.8119
Training time: 26.743321895599365s
```

Keras Tensor Flow - Number of Models Ran: 24, Best Score: 0.89, Time: 8s

```
   model.fit(train_images, train_labels, epochs=8)
   test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

   print('\nTest accuracy:', test_acc)
```

```
Epoch 1/8
1875/1875 [==============================] - 1s 710us/step - loss: 0.0991 - accuracy: 0.9630
Epoch 2/8
1875/1875 [==============================] - 1s 688us/step - loss: 0.1014 - accuracy: 0.9617
Epoch 3/8
1875/1875 [==============================] - 1s 679us/step - loss: 0.0954 - accuracy: 0.9643
Epoch 4/8
1875/1875 [==============================] - 1s 685us/step - loss: 0.0964 - accuracy: 0.9634
Epoch 5/8
1875/1875 [==============================] - 1s 669us/step - loss: 0.0923 - accuracy: 0.9656
Epoch 6/8
1875/1875 [==============================] - 1s 684us/step - loss: 0.0927 - accuracy: 0.9653
Epoch 7/8
1875/1875 [==============================] - 1s 667us/step - loss: 0.0916 - accuracy: 0.9658
Epoch 8/8
1875/1875 [==============================] - 1s 690us/step - loss: 0.0899 - accuracy: 0.9673
313/313 - 0s - loss: 0.5226 - accuracy: 0.8908

Test accuracy: 0.8907999992370605
```

XGBoost - Number of Models Ran: 6, Best Score: 0.91, Time: 864.88s

```
start = time.time()
xgb_clf = XGBClassifier(n_estimators=500, n_jobs=-1, learning_rate=0.5, max_deth= 5, min_child_weight= 1, seed=0)
xgb_clf.fit(x_train,y_train)
stop = time.time()
print(xgb_clf.get_params)
print(f"Training time: {stop - start}s")
y_pred = xgb_clf.predict(x_test)
print("test accuracy")
print(accuracy_score(y_test, y_pred))
```

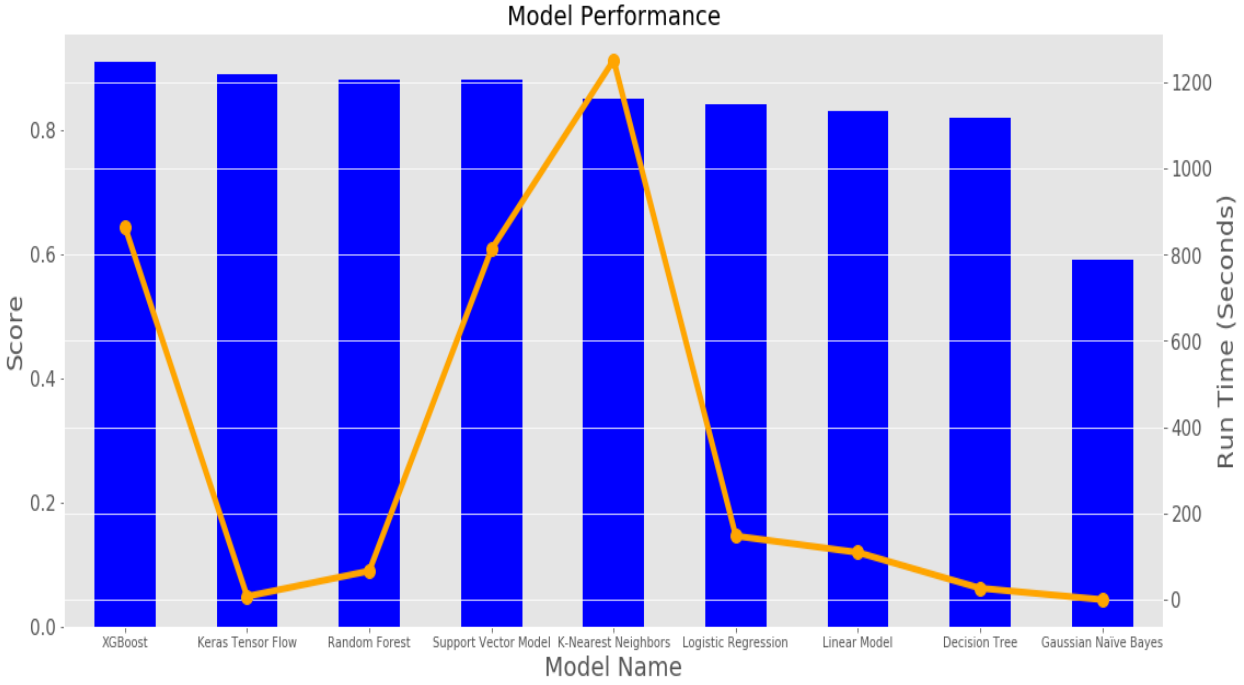
```
[11:21:56] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:573:
Parameters: { "max_deth" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[11:21:58] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was cha
nged from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
<bound method XGBModel.get_params of XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.5, max_delta_step=0, max_depth=6, max_deth=5,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=500, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, seed=0, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)>
Training time: 864.883885383606s
test accuracy
0.9053
```

Model Evaluation

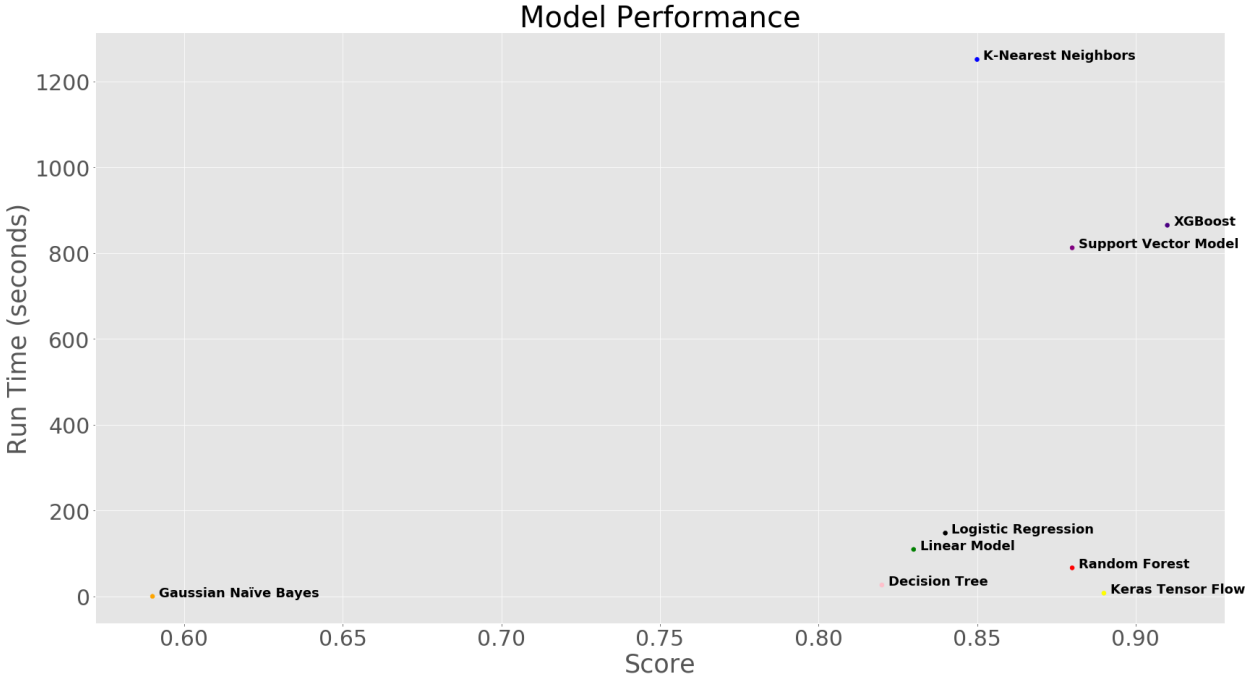
Model score and run time table sorted by score

Model Name	Score	Run Time
XGBoost	0.91	864.88
Keras Tensor Flow	0.89	8.00
Random Forest	0.88	66.95
Support Vector Model	0.88	812.27
K-Nearest Neighbors	0.85	1250.92
Logistic Regression	0.84	147.96
Linear Model	0.83	109.85
Decision Tree	0.82	27.00
Gaussian Naïve Bayes	0.59	0.54

Bar chart for scores with overlaying line plot for run times



Scatter plot of score vs. run Time



Conclusion and Interpretation of Results

This project was an interesting experience in using machine learning concepts for image recognition. Also, it confirmed that machine learning is in fact a good application for image recognition in the fashion industry. Majority of the models used were able to predict each image at an accuracy over 80%. Also, the run times for the models was not excessive considering the vast number of images in the data sets.

The best performing model by prediction score was XGBoost with 91% accuracy followed by Keras Tensorflow at 89%. For run time the Naïve Bayes model performed the best at only 0.54 seconds followed by Keras Tensorflow at 8 seconds. When factoring in process time and accuracy the Keras Tensorflow should be used due to its relatively high accuracy and low run time. If run time is not a concern than XGBoost should be considered due to its higher accuracy.

Overall, this project was valuable in gaining knowledge about computer vision and image recognition. Also, it was helpful for getting additional experience with many different machine learning techniques. We were even able to utilize models that were not covered in class like XGBoost and Keras Tensorflow. Going forward it would be interesting to look at using image recognition for other applications like healthcare or facial recognition.

Appendix

- GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark
- https://www.tensorflow.org/tutorials/keras/classification
- Kaggle - Fashion MNIST
- fashion-mnist/mnist_reader.py at master · zalandoresearch/fashion-mnist · GitHub
- https://iryndin.dev/post/xgboost_fashion_mnist/
- https://github.com/anktplwl91/fashion_mnist/blob/master/fashion_xgboost.ipynb
- https://www.kaggle.com/subhayan2018/exploring-xgboost-with-fashion-mnist
- http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf
- https://github.com/facebookresearch/faiss/blob/master/INSTALL.md
- https://towardsdatascience.com/make-knn-300-times-faster-than-scikit-learns-in-20-lines-5e29d74e76bb
- https://www.tensorflow.org/install/gpu