Fashion Classification

•••

2022.06.29

Team Members

Zihan Zhang
V00974121

Howard Luo
V00987151

Alex Huang
V00973109

What we do?

- **Introduce**
  - The fashion industry needs a viable way to efficiently and accurately classify images of fashion products in order to properly categorize the type of fashion item.
- **Datasets**
  - Kaggle Dataset
  - Fashion-Mnist
- **Methods**
  - RandomForest
  - CNN
  - ResNet
- Analysis
- Conclusion

Datasets

Kaggle Dataset

1. More than 5000 images with 20 different classes.
2. These images were contributed by 32 individuals worldwide who captured photos of their clothingitems and provided corresponding labels.

Fashion-Mnist

1. contain 70,000 clothing in 10 different categories.

2. All of the pictures are 28x28 gray-scale image.

	image	sender_id	label	kids
0	4285fab0-751a-4b74-8e9b-43af05deee22	124	Not sure	False
1	ea7b6656-3f84-4eb3-9099-23e623fc1018	148	T-Shirt	False
2	00627a3f-0477-401c-95eb-92642cbe078d	94	Not sure	False
3	ea2ffd4d-9b25-4ca8-9dc2-bd27f1cc59fa	43	T-Shirt	False
4	3b86d877-2b9e-4c8b-a6a2-1d87513309d0	189	Shoes	False
...	...	...	...	...
5398	dfd4079d-967b-4b3e-8574-fbac11b58103	204	Shorts	False
5399	befa14be-8140-4faf-8061-1039947e329d	204	Body	True
5400	5379356a-40ee-4890-b416-2336a7d84061	310	Shorts	False
5401	65507fb8-3456-4c15-b53e-d1b03bf71a59	204	Shoes	False
5402	32b99302-cec7-4dec-adfa-3d4029674209	204	Skirt	False

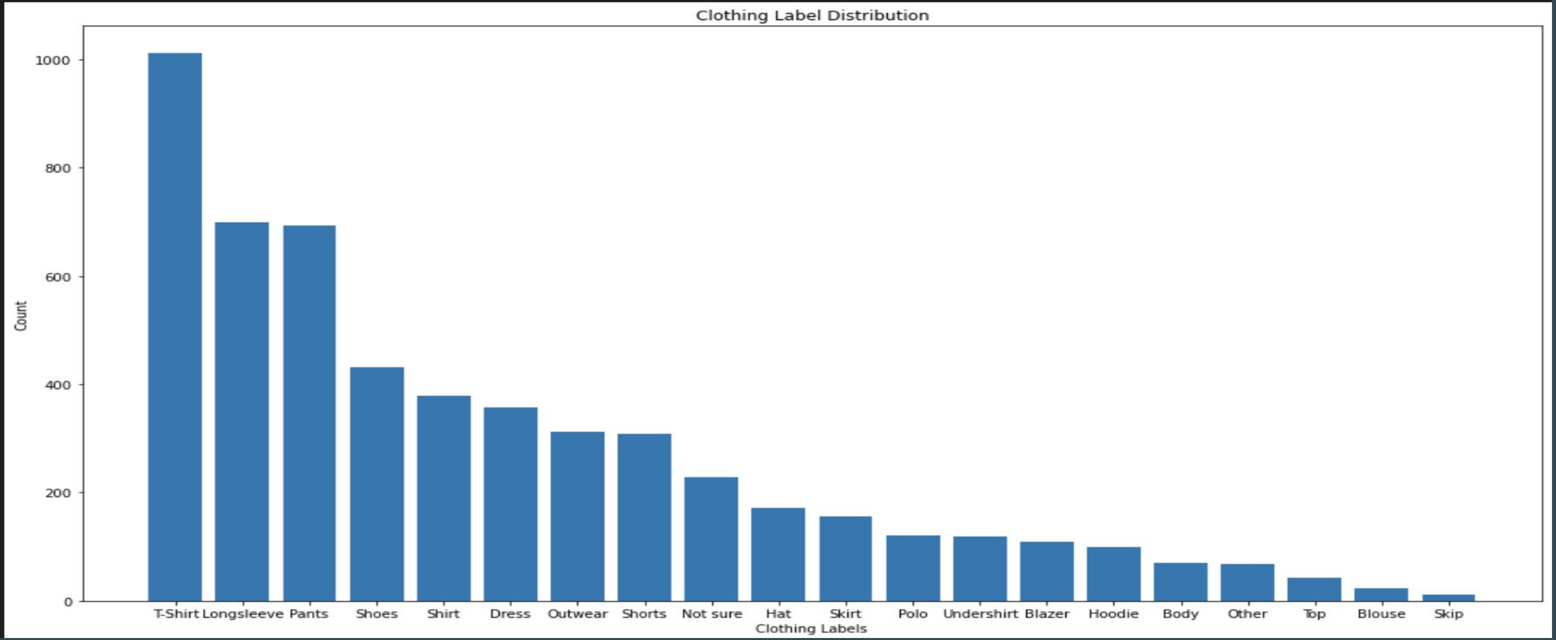
```
df["label"].value_counts()
```

✓ 0.1s

```
T-Shirt       1011
Longsleeve     699
Pants          692
Shoes          431
Shirt          378
Dress          357
Outwear        312
Shorts         308
Not sure       228
Hat            171
Skirt          155
Polo           120
Undershirt     118
Blazer         109
Hoodie         100
Body            69
Other           67
Top             43
Blouse          23
Skip            12
```

Kaggle Datasets

```python
import matplotlib.pyplot as plt

label_counts = clothing['label'].value_counts()

plt.figure(figsize=(18, 10))
plt.bar(label_counts.index, label_counts.values)
plt.xlabel('Clothing Labels')
plt.ylabel('Count')
plt.title('Clothing Label Distribution')
# plt.xticks(rotation=90)
plt.show()
```



Kaggle Dataset

Data Preprocess

Read Image

```python
#preprocess(read pic)
h = 28;w = 28;x = [];y = []
for f in files:
    img = cv2.imread(path+'/'+f, cv2.IMREAD_GRAYSCALE)
    pic = f.replace('.jpg', '')
    for i, label in enumerate(df1['image']):
        if label == pic:
            # background remove
            _, binary_img = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            segmented_img = cv2.bitwise_and(img, img, mask=binary_img)
            resized_img = cv2.resize(segmented_img, (h, w))
            x.append(resized_img.flatten())
            y.append(df1.loc[i, 'label'])  # Assuming df1 is a DataFrame with 'label' column
x = np.array(x)
y = np.array(y)
```

✓ 5m 52.1s

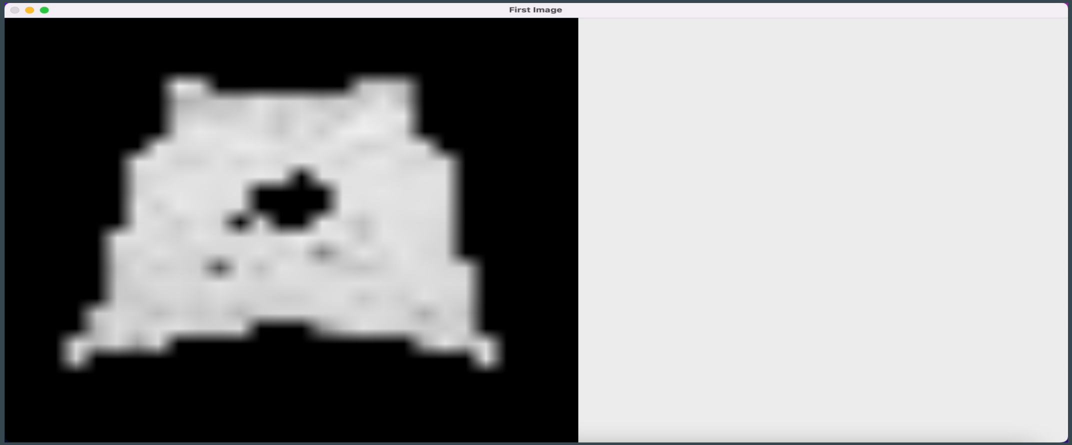
```python
# Horizontal rotate
flip_x = []
for image in x:
    flipped_image = cv2.flip(image, 1)

    flip_x.append(flipped_image.reshape(-1))

flipped_x = np.array(flip_x)
```

✓ 0.1s

- Use CV2 Read image and grayscale
- Use Background Removal function remove the image background, make the dataframe more clear
- Append image dataframe to X
- Appen image label to Y

- Then use CV2.flip(image,1), flip the dataframe (“1” means horizontal flip)
- Then append it to the flip_x[ ]

It can create more data as the training set for model, make the accurracy higher.



Orignal Image

Rebuild from Dataframe

```python
Merge_data = data + flip_data
Merge_data_label = data_label + flip_data_label

Merge_data = np.array(Merge_data)
Merge_data_label = np.array(Merge_data_label)
```

```python
#split the data into train and test
train_x,test_x,train_y,test_y=train_test_split(Merge_data, Merge_data_label, test_size = 0.2)
```

- Combine two array together
- split it to the train_set and Test_set

Methods

RandomForeast

–Kaggle dataset

Parameter "GINI"

```
model = RandomForestClassifier(n_estimators=100,criterion="gini" ,random_state=99)
model.fit(train_x,train_y)

y_pred = model.predict(test_x)

accuracy = accuracy_score(test_y,y_pred)
print("accuracy", accuracy)
```

✓ 6.5s

```
accuracy 0.9265463917525774
```

Parameter "entropy"

```
model = RandomForestClassifier(n_estimators=100,criterion="entropy" ,random_state=99)
model.fit(train_x,train_y)

y_pred = model.predict(test_x)

accuracy = accuracy_score(test_y,y_pred)
print("accuracy", accuracy)
```

✓ 9.1s

```
accuracy 0.9278350515463918
```

RandomForeast

–Kaggle and Fashion-Mnist Combined

Parameter "GINI"

```
model = RandomForestClassifier(n_estimators=100,criterion="gini", random_state=0)
model.fit(train_x, train_y)
# Predict the labels for the test set
y_pred = model.predict(test_x)
# Calculate accuracy
accuracy = accuracy_score(test_y, y_pred)
print('Accuracy:', accuracy)
```

✓ 11.5s

```
Accuracy: 0.9089347079037801
```

Parameter "entropy"

```
model = RandomForestClassifier(n_estimators=100,criterion="entropy", random_state=0)
model.fit(train_x, train_y)
# Predict the labels for the test set
y_pred = model.predict(test_x)
# Calculate accuracy
accuracy = accuracy_score(test_y, y_pred)
print('Accuracy:', accuracy)
```

✓ 17.7s

```
Accuracy: 0.9072164948453608
```

Convolutional Neural Network (CNN) –Kaggle dataset

The code to implement CNN :

```
# Building the CNN model
model = keras.Sequential([
    layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(label_encoder.classes_), activation='softmax')
])

# Compiling the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])

# Training the model
model.fit(augmented_train_images, augmented_train_labels, epochs=10, batch_size=32, validation_data=(test_images, test_labels))

# Evaluating the model
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
print('Test Loss:', test_loss)
print('Test Accuracy:', test_accuracy)
```

Convolutional Neural Network (CNN) –Kaggle dataset

No Pre-process data

Code

```
# Set the path for the image folder and CSV file
image_folder = 'images_original'
csv_file = 'images.csv'

# Load the CSV file
df = pd.read_csv(csv_file)
image_names = df['image'].values
labels = df['label'].values

# Encode the labels numerically
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)

# Split the dataset into training and testing sets
train_image_names, test_image_names, train_labels, test_labels = train_test_split(
    image_names, labels_encoded, test_size=0.2, random_state=42)

def load_and_preprocess_image(image_path):
    image = Image.open(image_path)
    image = image.resize((64, 64))  # resize to match the model input size
    if image.mode != 'RGB':  # if the image is grayscale
        image = image.convert('RGB')  # convert it to RGB
    image = np.array(image) / 255.0  # normalize pixel values to [0, 1]
    return image

# Load and preprocess the images for training and testing datasets
train_images = np.array([load_and_preprocess_image(os.path.join(image_folder, name+'.jpg')) for name in train_image_names])
test_images = np.array([load_and_preprocess_image(os.path.join(image_folder, name+'.jpg')) for name in test_image_names])
```

```
Epoch 1/10
136/136 [==============================] - 9s 62ms/step - loss: 2.4757 - accuracy: 0.2578 - val_loss: 2.2624 - val_accuracy: 0.3488
Epoch 2/10
136/136 [==============================] - 8s 61ms/step - loss: 2.0728 - accuracy: 0.4051 - val_loss: 2.1282 - val_accuracy: 0.3950
Epoch 3/10
136/136 [==============================] - 8s 59ms/step - loss: 1.7511 - accuracy: 0.5060 - val_loss: 1.9176 - val_accuracy: 0.4690
Epoch 4/10
136/136 [==============================] - 8s 60ms/step - loss: 1.4697 - accuracy: 0.5757 - val_loss: 1.8324 - val_accuracy: 0.4847
Epoch 5/10
136/136 [==============================] - 8s 61ms/step - loss: 1.2382 - accuracy: 0.6317 - val_loss: 1.7371 - val_accuracy: 0.5227
Epoch 6/10
136/136 [==============================] - 8s 60ms/step - loss: 1.0129 - accuracy: 0.7057 - val_loss: 1.7470 - val_accuracy: 0.5254
Epoch 7/10
136/136 [==============================] - 8s 60ms/step - loss: 0.8222 - accuracy: 0.7580 - val_loss: 1.8624 - val_accuracy: 0.5217
Epoch 8/10
136/136 [==============================] - 8s 59ms/step - loss: 0.6633 - accuracy: 0.8135 - val_loss: 1.9803 - val_accuracy: 0.5032
Epoch 9/10
136/136 [==============================] - 8s 60ms/step - loss: 0.5125 - accuracy: 0.8619 - val_loss: 1.9329 - val_accuracy: 0.5116
Epoch 10/10
136/136 [==============================] - 8s 60ms/step - loss: 0.3981 - accuracy: 0.8968 - val_loss: 2.0481 - val_accuracy: 0.5190
34/34 [==============================] - 1s 14ms/step - loss: 2.0481 - accuracy: 0.5190
Test Loss: 2.0481319427490234
Test Accuracy: 0.5189639329910278
```

Accuracy

Convolutional Neural Network (CNN) –Kaggle dataset

Pre-process data

```
# Set the tags to be kept
desired_labels = ['T-Shirt', 'Pants', 'Longsleeve', 'Dress', 'Outwear', 'Shirt', 'Shoes']
```
✓ 0.0s

```
# Filter data by tags to be retained
desired_indices = [i for i, label in enumerate(labels) if label in desired_labels]
desired_image_names = image_names[desired_indices]
desired_labels = labels[desired_indices]
```
✓ 0.0s

```
25/25 [==============================] - 0s 14ms/step - loss: 1.3192 - accuracy: 0.6572
Test Loss: 1.3192273378372192
Test Accuracy: 0.657216489315033
```

Accuracy

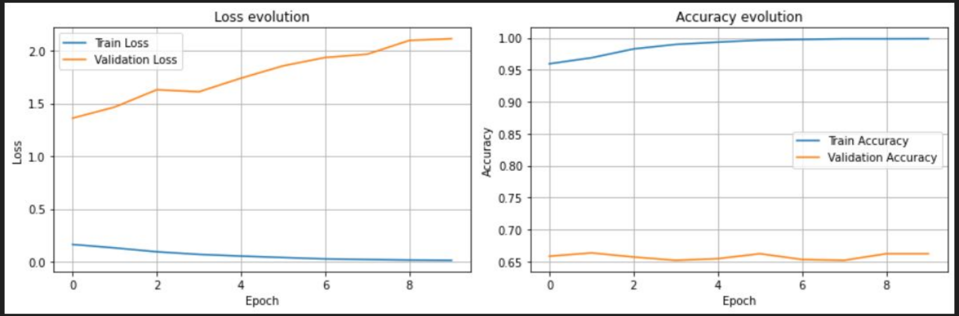
Convolutional Neural Network (CNN) –Kaggle dataset

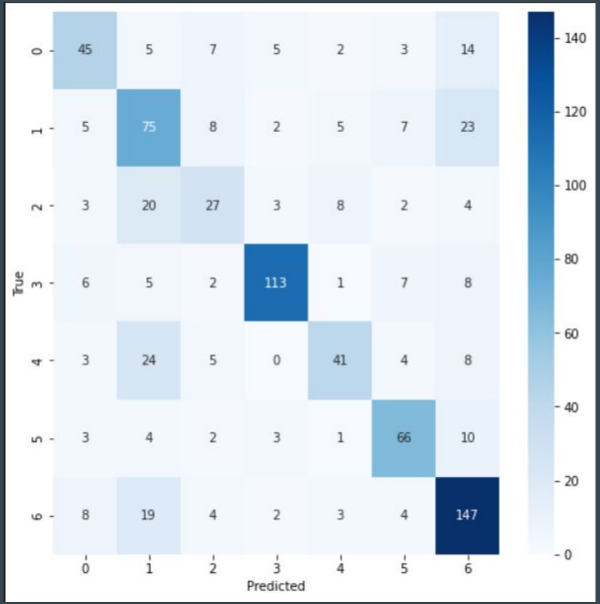
Inverting images to increase training samples

```python
for image, label in zip(train_images, train_labels):
    augmented_train_images.append(image)
    augmented_train_labels.append(label)
    # flip the images
    augmented_image = datagen.apply_transform(image, {'flip_horizontal': True})
    augmented_train_images.append(augmented_image)
    augmented_train_labels.append(label)
```

result :

```
Test Loss: 2.114187717437744
Test Accuracy: 0.6623711585998535
```





ResNet

```
# Classification head
x = Flatten()(block3_output)
x = Dense(128, activation='relu')(x)
predictions = Dense(len(np.unique(train_y)), activation='softmax')(x)

# Create and compile model
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer=Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Reshape data to match the input shape of the model
train_x = train_x.reshape(-1, 28, 28, 1)
test_x = test_x.reshape(-1, 28, 28, 1)

# Train the model
model.fit(train_x, train_y, epochs=5, validation_data=(test_x, test_y))
```

```
Epoch 5/5
194/194 [==============================] - 5s 26ms/step - loss: 0.0429 - accuracy: 0.9918 - val_loss: 0.4782 - val_accuracy:
0.9195
```

Resnet Accuracy

	train and test kaggle	Combined
RandomForest	0.925	0.908
CNN	0.65	0.69
ResNet	0.99	0.5128

Analysis and Conclusion

- RandomForest:
  RandomForest demonstrated good performance in classifying fashion products, especially when trained on a single dataset.
- CNN:
  Although the accuracy of the CNN model did not reach the level of RandomForest, it showed potential for fashion product classification when applied with appropriate preprocessing and data augmentation techniques.
- ResNet:shows strong dataset-specific performance but limited generalization to new data. Overall, the model’s accuracy is good to excellent, its ability to generalize is moderate, and its adaptability to new data ranges from moderate to low.