BÁO CÁO ĐỒ ÁN A.I

CHƯƠNG 1: Lựa chọn và Xây dựng Dataset (30%)

1.1. Bộ dữ liệu:

- Bộ dữ liệu được sử dụng ở đây là MNIST-Fashion, bao gồm 60.000 hình ảnh trong trainset và 10.000 hình ảnh trong test set
- Bộ dữ liệu này bao gồm hình ảnh của 10 classes (T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle Boot).
- Trong đồ án này, chỉ có dữ liệu từ trainset được sử dụng, trong đó 80% dữ liệu (48.000 ảnh) được sử dụng để tạo ra một bộ trainset mới (train_images) và 20% còn lại (12.000 ảnh) được sử dụng để tạo ra testset (test_images).

1.2. Tăng cường dữ liệu (Data Augmentation):

- Kĩ thuật lật ảnh (Flipping) được áp dụng cho Data Augmentation. Sẽ lựa chọn ảnh để lật theo trục ngang với tỉ lệ lật ảnh 50% và gán nhãn (label) 1 cho ảnh lật và 0 cho ảnh không lật.
- Cuối cùng, bộ trainset mới sẽ bao gồm cả trainset với ảnh không lật và trainset với ảnh đã lật .

CHƯƠNG 2: Kiến trúc mô hình:

Framework ***Tensorflow*** và ***Scikit-learn*** được áp dụng:

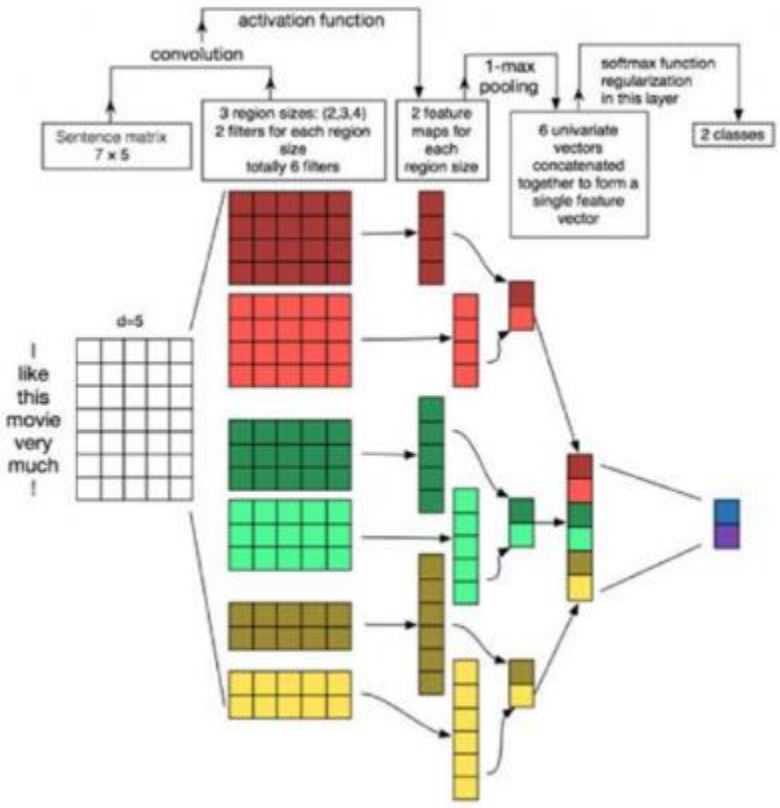
2.1. Convolutional Neural Network (CNN):

2.1.1. Định nghĩa:

CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng rộng rãi trong các bài toán xử lý ảnh và video. CNN hoạt động bằng cách học các đặc trưng trực quan từ dữ liệu đầu vào, sau đó sử dụng các đặc trưng này để phân loại, nhận diện hoặc dự đoán các đối tượng trong ảnh hoặc video.

Cấu trúc có 3 phần chính:

- ✓ **-Local receptive field** (trường cục bộ): Có nhiệm vụ lọc dữ liệu, hình ảnh, lựa chọn các vùng ảnh có giá trị sử dụng cao nhất
- ✓ **Shared weights and bias** (trọng số chia sẻ): giảm tối đa lượng tham số có tác dụng chính của yếu tố này trong mạng CNN.
- ✓ **Pooling layer** (lớp tổng hợp): Làm đơn giản các thông tin đầu ra. Có nghĩa là, sau khi đã hoàn tất tính toán và quét qua các lớp thì đến pooling layer để lược bớt các thông tin không cần thiết.



Cách chọn tham số cho CNN:

Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể

Filter size: thông thường filter theo size 5×5 hoặc 3×3

Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn

Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất.

2.1.2. Tại sao nên sử dụng mô hình này

Mô hình Convolutional Neural Network (CNN) có nhiều ưu điểm khi xử lý dữ liệu hình ảnh, bao gồm:

1. **Tính không gian (Spatial hierarchy)**: CNN có khả năng tự học và nhận biết các đặc trưng phức tạp từ hình ảnh bằng cách xây dựng một hệ thống phân cấp. Điều này giúp cho việc nhận dạng các đối tượng trong hình ảnh trở nên hiệu quả hơn.
2. **Giảm số lượng tham số**: So với các mô hình neural network truyền thống, CNN giảm đáng kể số lượng tham số cần học. Điều này giúp giảm bộ nhớ cần thiết để lưu trữ mô hình và tăng tốc độ huấn luyện.
3. **Bảo tồn tính không gian của hình ảnh**: CNN giữ được thông tin về vị trí và bối cảnh của các đặc trưng trong hình ảnh, điều mà các mô hình fully connected không thể làm được.
4. **Tự động học đặc trưng**: Không giống như các phương pháp truyền thống, CNN có khả năng tự động học các đặc trưng từ dữ liệu, giảm bớt công việc của con người trong việc chọn lọc và xây dựng đặc trưng.
   Vì những lý do trên, CNN thường được sử dụng rộng rãi trong các bài toán liên quan đến hình ảnh như nhận dạng khuôn mặt, phân loại hình ảnh, phát hiện đối tượng, và nhiều hơn nữa

2.2. Random forest classifier:

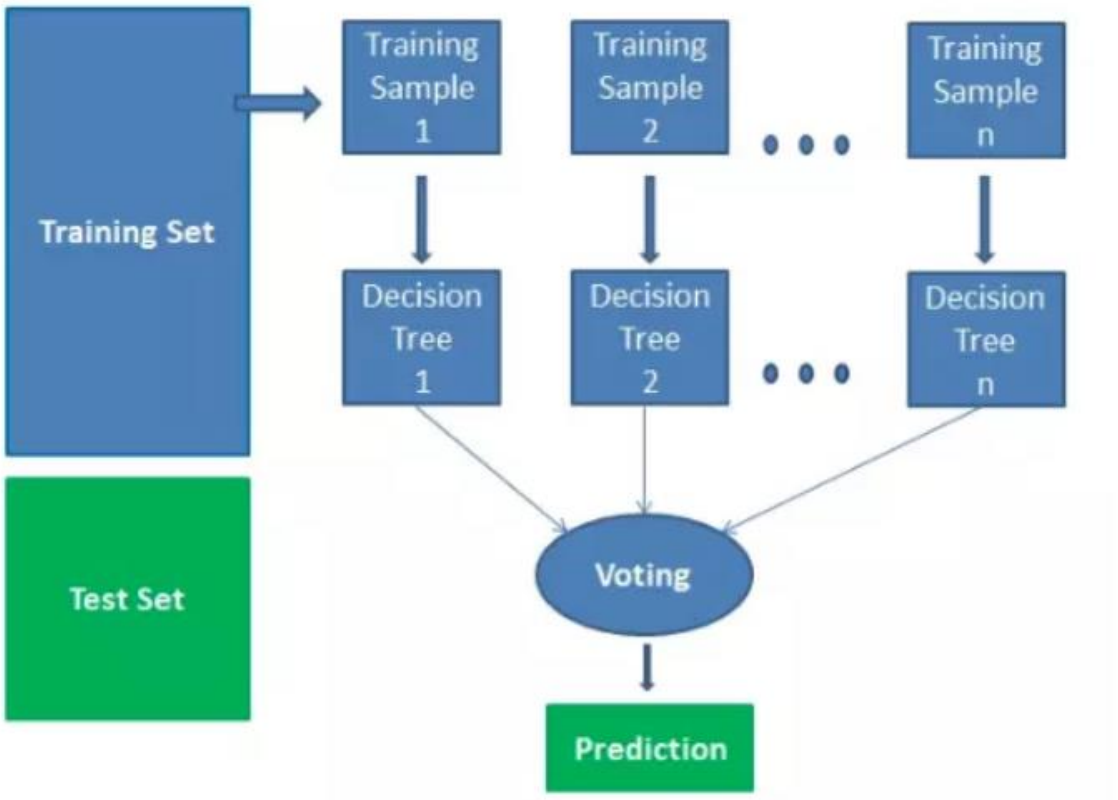
2.2.1. Định nghĩa:

Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Chẳng hạn như : một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh. Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. Nó cũng cung cấp một chỉ số khá tốt về tầm quan trọng của tính năng. Random forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng.

Cách hoạt động:

Nó hoạt động theo bốn bước:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Hãy bỏ phiếu cho mỗi kết quả dự đoán.
4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



2.2.2. Tại sao chúng ta nên sử dụng mô hình này:

Mô hình Random Forest có nhiều ưu điểm khi xử lý dữ liệu, bao gồm:

1. Hiệu suất cao: Random Forest thường cho kết quả dự đoán chính xác và ổn định, không yêu cầu nhiều tinh chỉnh tham số và không dễ bị overfitting như một số mô hình khác.
2. Xử lý được dữ liệu phức tạp: Random Forest có thể xử lý dữ liệu với nhiều loại biến (liên tục, phân loại), và không yêu cầu giả định về phân phối của dữ liệu.
3. Đánh giá tầm quan trọng của các biến: Random Forest cung cấp một chỉ số để đánh giá mức độ quan trọng của từng biến đối với việc dự đoán kết quả. Điều này rất hữu ích khi cần hiểu rõ hơn về mô hình và dữ liệu.
4. Khả năng xử lý dữ liệu lớn: Random Forest có thể xử lý dữ liệu với số lượng mẫu lớn và nhiều biến một cách hiệu quả.
5. Khả năng chịu nhiễu tốt: Random Forest có khả năng chịu được nhiễu trong dữ liệu, như dữ liệu bị thiếu hoặc nhiễu.

Chương 3: Đánh giá mô hình:

3.1. Các thử nghiệm và kết quả đạt được:

Với CNN:

- **Learning Rate Schedule:**
  - tf.keras.optimizers.schedules.ExponentialDecay: Gồm lr_schedule mà tốc độ học giảm theo một tốc độ mũ với mỗi bước (step) trong quá trình huấn luyện.
  - initial_learning_rate: Là tốc độ học ban đầu. Trong trường hợp này, tốc độ học ban đầu là 0.0005.
  - decay_steps: Là số bước mà mỗi mức giảm tốc độ học sẽ diễn ra. Trong trường hợp này, mỗi 1000 bước tốc độ học sẽ giảm.
  - decay_rate: Là tỷ lệ giảm tốc độ học sau mỗi decay_steps. Trong trường hợp này, tốc độ học sẽ giảm đi 10% sau mỗi 1000 bước.
  - staircase: Là một biến để xác định decay_steps có nên được sử dụng theo cách bước nguyên (True) hay không (False). Trong trường hợp này, staircase là True, nghĩa là decay_steps sẽ được sử dụng theo cách bước nguyên.

- **Define model:**
  - Hai lớp Conv2D với 32 filters với kích thước kernel là 3x3 (kernel_size=3) và hàm kích hoạt (activation) là ReLU (Rectified Linear Activation).
  - Một lớp MaxPooling2D với kích thước cửa sổ là 2x2 giúp giảm số lượng tham số và tính toán trong mạng CNN.
  - Một lớp Flatten để chuyển đổi feature maps thành vector 1 .
  - Một lớp Dropout với tỷ lệ 0.5 để ngăn chặn overfitting.
  - Hai lớp Dense (hoàn toàn kết nối), với 128 neuron và hàm kích hoạt (activation) là ReLU cho lớp đầu tiên, và 10 neuron và hàm kích hoạt (activation) softmax cho lớp cuối cùng.

- **Training the Model:**
  - train_images_resized và train_labels: Là dữ liệu huấn luyện, gồm các hình ảnh đã được resize và label tương ứng.
  - epochs: Số lần lặp lại qua toàn bộ dữ liệu huấn luyện. Trong trường hợp này là 5.
  - batch_size: Số lượng mẫu dữ liệu được sử dụng trong mỗi lần cập nhật trọng số mô hình. Trong trường hợp này, batch_size được đặt là 64.
  - validation_data: Dữ liệu sử dụng để đánh giá hiệu suất của mô hình sau mỗi epoch, trong trường hợp này là test_images_resized và test_labels.

- **Compiling the Model:**
  - tf.keras.optimizers.Adam: Với tốc độ học (learning_rate)được cung cấp từ lịch trình tốc độ học (lr_schedule) đã được xác định trước.

  - loss: Hàm mất mát được sử dụng là 'sparse_categorical_crossentropy'
  - metrics: Là chỉ số được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Trong trường hợp này, sử dụng 'accuracy' để đo lường độ chính xác của mô hình.

Kết quả đạt được:

```
################################################
#  CNN prediction by passing per image test set  #
################################################
predicted_labels_CNN = []

for i in range(len(test_images_resized)):
    # Make prediction on a single image
    prediction = model.predict(np.expand_dims(test_images_resized[i], axis=0))
    # Add label after prediction into array
    predicted_label = np.argmax(prediction)
    predicted_labels_CNN.append(predicted_label)

print("Predicted labels of images:", predicted_labels_CNN)

# Calculate accuracy
accuracy = accuracy_score(test_labels, predicted_labels_CNN)
print("Accuracy:", accuracy)
```

✓ 16m 4.3s

```
Accuracy: 0.9068333333333334
```

Độ chính xác khoảng 90.68%.

Với Random Forest:

1. RandomForestClassifier(n_estimators=100, random_state=20): Định nghĩa một mô hình RandomForestClassifier với các tham số sau:

- n_estimators: Số lượng cây trong rừng. Trong trường hợp này, có 100 cây.
- random_state: Seed để thiết lập trạng thái ngẫu nhiên.

2. Trong khi RandomForestClassifier không yêu cầu dữ liệu ảnh được trình bày theo cấu trúc của một CNN, nó yêu cầu mỗi mẫu dữ liệu là một vector 1D thay vì một tensor đa chiều. Do đó, sử dụng tf.reshape để chuyển đổi dữ liệu ảnh từ tensor 4D (có kích thước (batch_size, 28, 28, 1)) thành tensor 2D (có kích thước (batch_size, 28*28)), sao cho mỗi ảnh được biểu diễn dưới dạng một vector 1D.

3. Huấn luyện mô hình RandomForestClassifier:

RF.fit(train_images_flattened, train_labels): Dữ liệu huấn luyện được truyền vào mô hình để huấn luyện. Trong trường hợp này, train_images_flattened là dữ liệu ảnh đã được làm phẳng và train_labels là nhãn tương ứng với các ảnh trong test set.

Kết quả đạt được:

```
# ################################################################
# # Random Forest prediction by passing per image in test set  #
# ################################################################

for i in range(len(test_images_flattened)):
    # Make prediction on a single image
    prediction = RF.predict(np.expand_dims(test_images_flattened[i], axis=0))
    # Add label after prediction into array
    predicted_label = prediction[0]
    predicted_labels_RF.append(predicted_label)

print("Predicted labels of images:", predicted_labels_RF)

# Calculate accuracy
accuracy_rf = accuracy_score(test_labels, predicted_labels_RF)
print("Accuracy:", accuracy_rf)
✓ 2m 5.5s
```

```
Accuracy: 0.98525
```

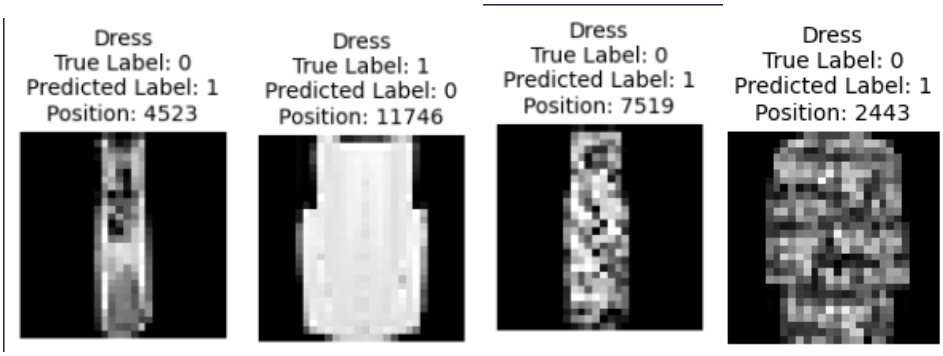
Độ chính xác khoảng 98.53%

3.2: Phân tích các trường hợp ảnh bị phân loại sai:

3 lớp dự đoán sai nhiều: Dress, Bag và Shirt.

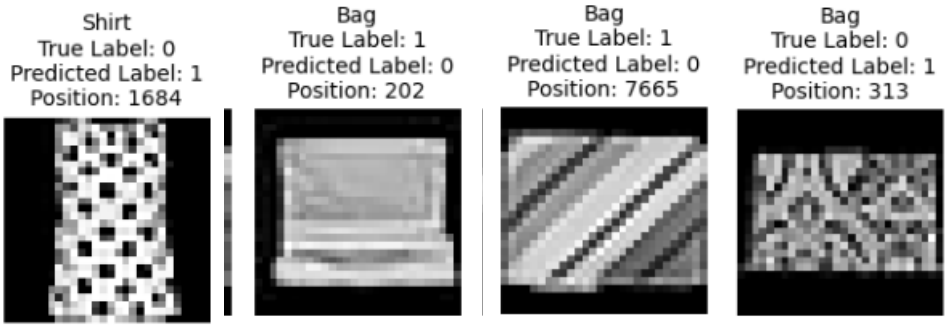
Ảnh bị dự đoán sai có thể vì những lý do sau:

1. **Ảnh với độ phân giải thấp:**



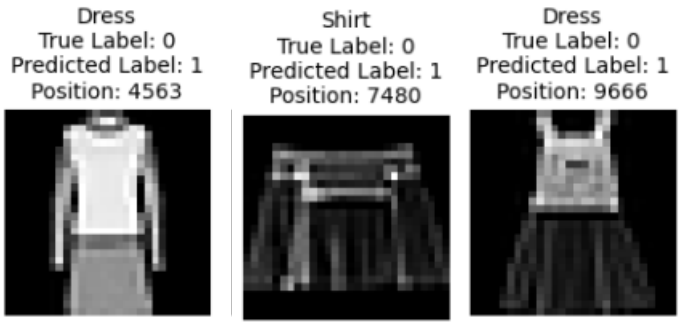
Những ảnh độ phân giải thấp (mờ) và đôi khi rất khó thấy, sẽ là những ảnh dễ bị dự đoán sai.

2. **Ảnh, lật và không lật, tương đối giống nhau:**



Có thể gây ra dự đoán sai do sự tương đồng giữa các phiên bản, đặc biệt là khi các chi tiết không thay đổi đáng kể sau khi ảnh được lật

3. **Mô hình học chưa đủ tốt:**



Mặc dù các ảnh có độ phân giải cao và chi tiết, nhưng mô hình vẫn phân loại sai do khả năng học chưa đủ tốt

THÀNH VIÊN

Trịnh Gia Tiến - 2159012

Bùi Thanh Tùng - 2159013

Phạm Nguyễn Gia Hưng - 2159022

Lê Trần Hiếu Nhân - 2159023