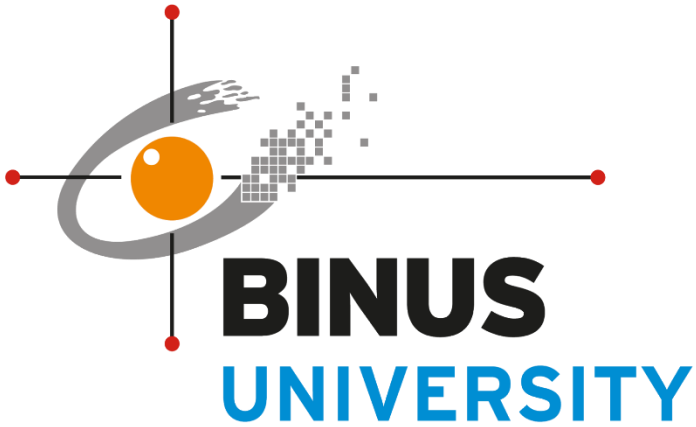
Project ANN “Image Classifier for Clothing Application using Convolutional Neural Network”



Anggota Kelompok
2301931251 - Daffa Rizki Rizaly
2301937293 - Edbert Khovey
2301869840 - William Yulio

School of Computer Science Program
Studi Teknik Informatika
Universitas Bina Nusantara
2020/2021

Idea Summary

Pada project Artificial Neural Network ini, kami ingin mengklasifikasikan gambar dari berbagai jenis foto pakaian kedalam kelasnya masing-masing. Kami memilih tema project ini, karena *clothes classification* dapat di implementasikan kedalam media sosial atau bisa juga digunakan sebagai teknik data gathering di suatu tempat. Sebagai contoh, dengan teknologi ini kita bisa secara otomatis mengklasifikasikan seorang user menyukai baju seperti apa, sehingga iklan yang ditampilkan ke user bisa disesuaikan berdasarkan data tersebut. Kami melihat bahwa penerapan model convolutional neural network itu dapat diaplikasikan kedalam banyak scenario, oleh karena itu kami memilih ide ini.

Dataset

Untuk dataset pada project ini berasal dari Github yaitu Fashion-MNIST (30 Mb) yang dibuat oleh Han Xiao, Kashif Rasul, dan Roland Vollgrafdan. Dataset ini terdiri dari *data training* yang terdiri dari 60,000 data dan *data testing* yang terdiri dari 10,000 data. Data-data ini merupakan kumpulan foto *grayscale* yang berukuran 28 * 28 dan terdiri dari 10 kelas. Inilah table dari kelas-kelas tersebut:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Berikut adalah link untuk datasetnya : [zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark (github.com)]

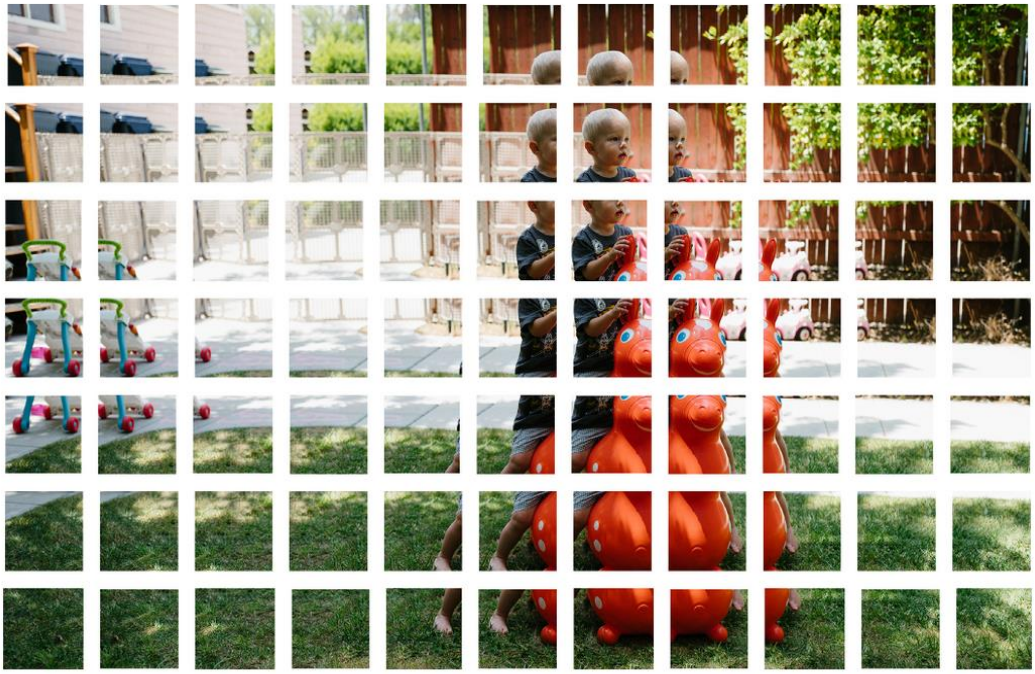
Literature Review

Convolutional Neural Network(CNN)

Convolutional Neural Network merupakan salah satu jenis neural network yang biasa digunakan untuk melakukan proses pengolahan data berupa gambar. Secara garis besar Convolutional Neural Network (CNN) tidak jauh beda dengan neural network biasanya. CNN terdiri dari neuron yang memiliki weight, bias dan activation function. Namun berbeda dengan ANN, CNN tidak menggunakan hidden layer tetapi menggunakan convolutional layer sebagai pengganti hidden layer. Convolutional layer juga terdiri dari

neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixels). Berikut adalah langkah-langkah bagaimana algoritma CNN dapat bekerja :

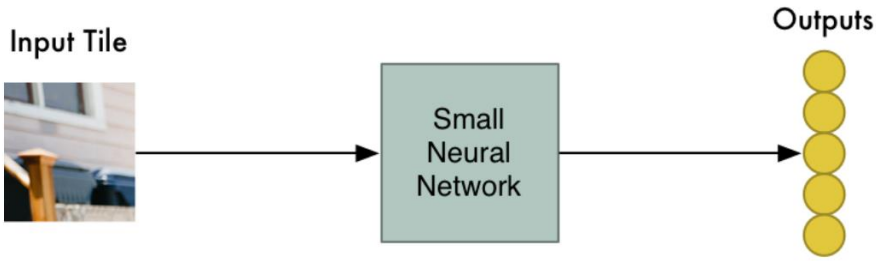
a. Memecahkan gambar menjadi bagian yang lebih kecil dan saling tumpang tindih



Source image : Adam Geitgey via Medium

Pada tahap ini, image yang sudah kita input ke dalam model akan masuk kedalam convolutional layer dan dilakukan proses pemecahan gambar menjadi bagian yang lebih kecil dan saling tumpang tindih. Convolutional layer merupakan sebuah layer yang berguna untuk melakukan *feature extraction* dari input gambar yang sudah kita masukkan kedalam model.

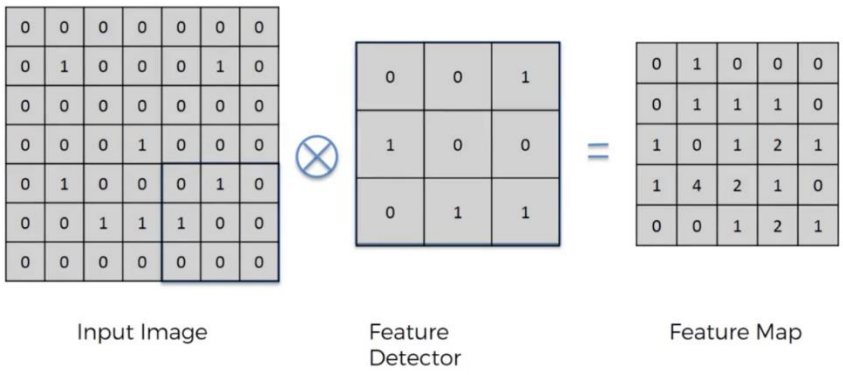
b. Memasukkan setiap gambar yang sudah dipecah sebelumnya kedalam small neural network (Filter layer)



Source image : Adam Geitgey via Medium

Setiap gambar yang sudah dipecah menjadi bagian bagian kecil tersebut, kemudian akan dimasukkan kedalam filter layer seperti pada gambar diatas. Proses ini dikenal dengan sebutan feature extraction, dimana feature extraction ini akan menghasilkan sebuah representasi feature dari gambar yang sudah dipecah sebelumnya. Output ini akan berguna bagi convolutional layer karena memberikan kemampuan mengenali object tersebut. Apabila terdapat object yang sama pada inputan yang berbeda, convolutional layer akan dengan cepat mengenali object tersebut.

Semua proses ini dilakukan pada masing-masing gambar kecil tersebut dengan menggunakan filter yang sama. Didalam convolutional layer mengenal adanya weight sharing, dimana setiap layer yang akan melakukan proses kalkulasi di dalam convolutional layer akan dikalikan dengan weight yang sama. Kemudian juga apabila terdapat bagian dari gambar yang tampaknya menarik, maka gambar tersebut akan ditandai sebagai *object of interest*. Hasil dari proses pada convolutional layer ini akan disimpan kedalam sebuah array bernama feature map. Di dalam convolutional layer akan dilakukan proses komputasi sebagai berikut :



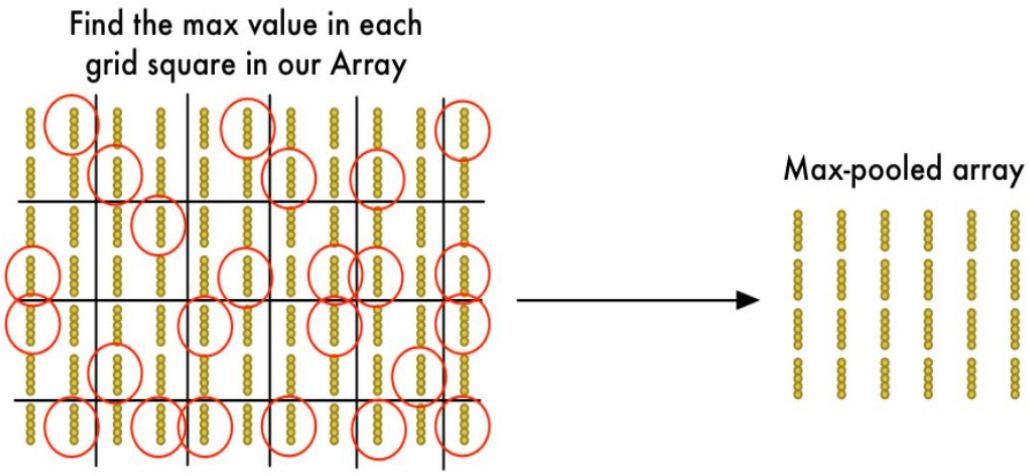
Gambar tersebut akan diubah kedalam bentuk matrix dan dilakukan operasi dot antara gambar dengan filter (feature detector). Proses tersebutlah yang akan menghasilkan feature map. Ada beberapa hal yang mempengaruhi hasil dari feature map ini, yaitu :

- Stride merupakan parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai stride adalah 1, maka layer filter akan bergeser sebanyak 1 pixel secara horizontal sampai dengan posisi matrix sudah berada di ujung maka, layer filter akan turun 1 pixel secara vertikal.
- Padding merupakan parameter menentukan jumlah pixel (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari convolutional layer (feature map).

c. Downsampling

Pada langkah sebelumnya, hasil dari proses convolution matrix akan disimpan kedalam array bernama feature map. Namun array tersebut memiliki ukuran yang terlalu besar dan akan membuat data menjadi sulit untuk diolah, oleh

karena itu kita akan melakukan proses downsampling untuk mengecilkan ukuran array. Proses ini akan melibatkan pooling layer, pooling layer sendiri merupakan layer terdiri dari sebuah filter dengan ukuran dan stride tertentu yang bergeser pada seluruh area feature map. Pooling yang biasa digunakan adalah Max Pooling dan Average Pooling. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus di update semakin sedikit dan mengatasi overfitting. Untuk proses downsampling ini, kita akan menggunakan max pooling atau mengambil nilai pixel terbesar dari setiap pooling kernel. Dengan begitu, sekalipun mengurangi jumlah parameter, informasi terpenting dari bagian tersebut tetap diambil. Di dalam pooling layer juga dikenal proses flattening, yaitu proses mengubah semua array 2 dimensi yang dihasilkan menjadi satu vektor linier kontinu panjang.



Source image : Adam Geitgey via Medium

d. Membuat prediksi

Setelah melewati semua proses itu, kita akan masuk ke bagian akhir yaitu memasukkan jaringan saraf lainnya dan melalui proses tersebut akan ditentukan apakah gambar tersebut cocok atau tidak dengan kelas dari label yang ada. Proses ini biasa disebut dengan fully-connected layer. Lapisan Fully-connected adalah lapisan dimana semua neuron aktivitas dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti hal nya jaringan syaraf tiruan bisa. Setiap aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan Fully-Connected.

Methodology

Link Code :
https://colab.research.google.com/drive/10HSgFZ4N6uMCy4hDydVKpjSHRwimyHt2?usp=sharing

1. Import Library

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import itertools
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
```

- Import tensorflow digunakan sebagai library machine learning yang digunakan untuk mengakses library keras.
- Import keras digunakan sebagai API untuk deep learning.
- Import matplotlib.pyplot digunakan untuk memvisualisasi data, hasil prediksi model dan semua yang butuh visualisasi grafik.
- Import Itertools digunakan untuk menggunakan product() yang digunakan untuk menghitung hasil cartesian.
- Import train_test_split digunakan untuk membagi dataset untuk data training dan data test.
- Import Confusion matrix dan classification report digunakan sebagai metode evaluasi model.
- Import Sequential digunakan untuk membuat model sequential sebagai model yang akan menampung berbagai layer seperti conv2d dan maxpooling2d.
- Import Dense digunakan untuk menambahkan layer yang fully-connected layer.
- Import Activation digunakan untuk menggunakan activation function seperti Relu dan sigmoid/softmax.
- Import Dropout digunakan untuk men-drop random neuron.
- Import Flatten digunakan untuk mengubah 2D array menjadi satu linear vector continue panjang.
- Import Conv2D digunakan untuk membuat convolutional layer.
- Import MaxPooling2D digunakan untuk mengurangi dimensi feature map.
- Import to_categorical digunakan untuk mengubah output yang categorical menjadi matrix dalam bentuk angka.
- Import Adam digunakan sebagai optimizer untuk model yang akan kami buat.

Kemudian setelah itu kami mengimport dataset dengan keras dataset fashion_mnist lalu kami memasukan dataset tersebut kedalam variabel x_train, y_train, x_test, dan y_test.

2. Data exploration

```
x_train = x_train / 255.0
x_test = x_test / 255.0

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

classes = {0: 'T-shirt/top',
           1: 'Trouser',
           2: 'Pullover',
           3: 'Dress',
           4: 'Coat',
           5: 'Sandal',
           6: 'Shirt',
           7: 'Sneaker',
           8: 'Bag',
           9: 'Ankle boot'}

class_names = ['T-shirt/top','Trouser','Pullover','Dress','Coat',
                       'Sandal','Shirt','Sneaker','Bag','Ankle boot']

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(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

Pada bagian ini, kami ingin melihat dataset yang kami gunakan seperti apa detail dan bentuknya. Pertama kami melakukan pembagian di variabel x_train dan x_test dibagi 255 untuk dinormalisasi data agar data dapat diolah dengan lebih mudah oleh sistem. Setelah itu, kami menampilkan heatmap dari data x_train[0] untuk melihat seperti apa bentuk datasetnya. Lalu dari data ini, kami bisa mengetahui bahwa dataset ini terdiri dari gambar berwarna grayscale. Selanjutnya di class_names kita membuat array yang berisikan nama kelas-kelas tersebut. Setelah itu kita memvisualisasikan untuk melihat pakaian apa saja yang ada.

3. Split Dataset

```
X_train, X_validate, y_train, y_validate = train_test_split(x_train, y_train, test_size = 0.2, random_state=42)
```

Kemudian kami membagi data x_train dan y_train untuk data validation sebesar 20% dan 80% data training dengan random state 42. Proses ini menggunakan train_test_split().

4. **Image Reshaping**

```
image_rows = 28
image_cols = 28
input_shape = (image_rows, image_cols, 1)

X_train = X_train.reshape(X_train.shape[0], image_rows, image_cols, 1)
X_test = x_test.reshape(x_test.shape[0], image_rows, image_cols, 1)
X_validate = X_validate.reshape(X_validate.shape[0], image_rows, image_cols, 1)

print('x_train shape: {}'.format(X_train.shape))
print('x_test shape: {}'.format(X_test.shape))
print('x_validate shape: {}'.format(X_validate.shape))
```

Selanjutnya kami melakukan image reshaping dengan menggunakan konfigurasi ini, dimana data direshape menjadi (jumlah data test/training, ukuran gambar secara baris (28), ukuran gambar secara kolom (28), dan 1 (karena dataset gambar yang digunakan hanya memiliki 1 warna yaitu grayscale)). Proses ini dilakukan agar data dapat diolah didalam model yang akan kita buat nantinya.

5. **To_categorical**

```
y_train = to_categorical(y_train, 10)
y_validate = to_categorical(y_validate, 10)
y_test = to_categorical(y_test, 10)
```

Pada tahap ini, semua label akan kita ubah yang semula dalam berupa string (kelas) menjadi kedalam bentuk angka.

6. **Membuat model**

```
cnn_model = Sequential()
```

Kemudian kami membuat model sequential supaya bisa tambahkan layer untuk membangun model CNN. Algoritma CNN sendiri terdiri dari convolutional layer, pooling layer, dan dense layer (fully-connected layer). Kemudian setelah membuat model sequential barulah kita bisa membuat model dengan multiple layer seperti CNN.

Penjelasan CNN 1 layer

```
cnn_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
#ada 32 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap detail image yang lebih bagus dibanding
5*5, dan keperluan komputasi 3*3 lebih kecil dibanding 5*5. memilih relu sebagai activation function karena relu
adalah default nya.

cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
#pooling mengurangi dimensionality dari featured map tapi tetap mempertahankan informasi yang ada di dalamnya,
pooling digunakan untuk mempercepat prose training, dan menghindari overfit.

cnn_model.add(Dropout(0.25))
#dropout untuk randomly shutdown neuron because kalau neuron udah kebanyakan biasnya itu isinya mirip atau bahakan
sama, jadinya dengan randomly shutdown neuron, kita bisa percepat learning dan hemat computer power.

cnn_model.add(Flatten())
#flatten adalah proses untuk mengubah data dari 2d array ke vector.

cnn_model.add(Dense(128, activation='relu'))
#dense = fully connected layer, fully connected layer mempunyai arti bahwa neuron di layer sebelumnya itu terhubung
dengan neuron di layer selanjutnya.

cnn_model.add(Dropout(0.2))

cnn_model.add(Dense(10, activation='softmax'))
#softmax function biasanya digunakan di output layer di clustering algorithm, karena output dari softmax ini selalu 1
atau lebih dari 1. tidak seperti sigmoid yang outputnya antara 0 dan 1
```

```
cnn_model.compile(optimizer= Adam(lr= 0.0001), # menggunakan adam karena adam paling cepat dan paling bagus
          loss='categorical_crossentropy',
          metrics=['accuracy'])#categorical cross entropy mengukur performance dari model berdasarkan output yang
                               berada di antara 0 dan 1. loss bertambah jika predicted probability melenceng dari label
```

Penjelasan CNN 2 layer

```
cnn_model_2.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
#ada 32 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap detail image yang lebih bagus dibanding
5*5, dan keperluan nkomputasi 3*3 lebih kecill dibanding 5*5. memilih relu sebagai activation function karena relu
adalah default nya.

cnn_model_2.add(Dropout(0.2))
#dropout untuk randomly shutdown neuron because kalau neuron udah kebanyakan biasnya itu isinya mirip atau bahakan
sama, jadinya dengan randomly shutdown neuron, kita bisa percepat learning dan hemat computer power.

cnn_model_2.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
#ada 64 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap detail image yang lebih bagus dibanding
5*5, dan keperluan nkomputasi 3*3 lebih kecill dibanding 5*5. memilih relu sebagai activation function karena relu
adalah default nya.
#karena namanya CNN 2 ya artinya jadi ada 2 layer cnn nya deh

cnn_model_2.add(MaxPooling2D(pool_size=(2, 2)))
#pooling mengurangi dimensionality dari featured map tapi tetap mempertahankan informasi yang ada di dalamnya,
pooling digunakan untuk mempercepat prose training, dan menghindari overfit.


cnn_model_2.add(Dropout(0.3))
cnn_model_2.add(Flatten())
#flatten adalah proses untuk mengubah data dari 2d array ke vector.


cnn_model_2.add(Dense(64, activation='relu'))
#dense = fully connected layer, fully connected layer mempunyai arti bahwa neuron di layer sebelumnya itu terhubung
dengan neuron di layer selanjutnya.
model.add(Dropout(0.2))
cnn_model_2.add(Dense(128, activation='relu'))
#dense ada 2 karena namanya CNN 2layer jadinya ada 2 layer ya dense nya harus ditambahin juga
model.add(Dropout(0.2))

cnn_model_2.add(Dense(10, activation='softmax'))
#softmax function biasanya digunakan di output layer di clustering algorithm, karena softmax ini merubah raw value
menjadi sesuatu yang pasti. jumlah dari output probabilities di fully connected layer itu selalu 1.
```

Penjelasan CNN 3 layer

```
cnn_model_3.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=input_shape))
#ada 32 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap
detail image yang lebih bagus dibanding 5*5, dan keperluan nkomputasi 3*3
lebih kecill dibanding 5*5. memilih relu sebagai activation function karena
relu adalah default nya.
cnn_model_3.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
#ada 64 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap
detail image yang lebih bagus dibanding 5*5, dan keperluan nkomputasi 3*3
lebih kecill dibanding 5*5. memilih relu sebagai activation function karena
relu adalah default nya.
#karena namanya CNN 3 ya artinya jadi ada 3 layer cnn nya deh
cnn_model_3.add(MaxPooling2D(pool_size=(2, 2)))
#pooling mengurangi dimensionality dari featured map tapi tetap
mempertahankan informasi yang ada di dalamnya, pooling digunakan untuk
mempercepat prose training, dan menghindari overfit.
cnn_model_3.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
#ada 128 filter, dengan 1 filternya berukuran 3*3, karena 3*3 bisa menangkap
detail image yang lebih bagus dibanding 5*5, dan keperluan nkomputasi 3*3
lebih kecill dibanding 5*5. memilih relu sebagai activation function karena
relu adalah default nya.
#karena namanya CNN 3 ya artinya jadi ada 3 layer cnn nya deh
cnn_model_3.add(MaxPooling2D(pool_size=(2, 2)))
#pooling mengurangi dimensionality dari featured map tapi tetap
mempertahankan informasi yang ada di dalamnya, pooling digunakan untuk
mempercepat prose training, dan menghindari overfit.

cnn_model_3.add(Dropout(0.3))#dropout untuk randomly shutdown neuron because
kalau neuron udah kebanyakan biasnya itu isinya mirip atau bahakan sama,
jadinya denga randomly shutdown neuron, kita bisa percepat learning dan hemat
computer resources
cnn_model_3.add(Flatten()) #flattening untuk bisa dimasukkan ke NN
#flatten adalah proses untuk mengubah data dari 2d array ke vector.

cnn_model_3.add(Dense(128, activation='relu'))
#dense = fully connected layer, fully connected layer mempunyai arti bahwa
neuron di layer sebelumnya itu terhubung dengan neuron di layer selanjutnya.
model.add(Dropout(0.2))

cnn_model_3.add(Dense(256, activation='relu'))
#dense = fully connected layer, fully connected layer mempunyai arti bahwa
neuron di layer sebelumnya itu terhubung dengan neuron di layer selanjutnya.
model.add(Dropout(0.2))

cnn_model_3.add(Dense(10, activation='softmax'))#softmax function is used to
normalize the outputs, converting them from weighted sum values into
probabilities that sum to one
```

```
history = cnn_model.fit(X_train, y_train,
          batch_size=300,
          epochs=75,
          verbose=1,
          validation_data=(X_validate, y_validate))
```

Setiap selesai melakukan kalkulasi pada model CNN dengan layer 1, 2, dan 3, kemudian melakukan proses fitting dengan data training lalu menggunakan data validasi untuk melakukan proses testing.

```
plt.figure(figsize = (18,8))
plt.subplot(121)
plt.plot(history.history['loss'], color='darkcyan', label="Training loss")
plt.plot(history.history['val_loss'], color='maroon', label="validation
loss",)
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend(loc='best', shadow=True)
plt.title('1-Convolution Layer Loss')
plt.subplot(122)
plt.plot(history.history['accuracy'], color='darkcyan', label="Training
accuracy")
plt.plot(history.history['val_accuracy'], color='maroon',label="Validation
accuracy")
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend(loc='best', shadow=True)
plt.title('1-Convolution Layer Accuracy')
plt.show()
```

Setelah selesai melakukan kalkulasi disetiap layer CNN, melakukan proses training, dan validasi maka hasilnya akan di plot kedalam bentuk grafik untuk melihat apakah hasil loss function dan accuracy dari data yang ditraining dan data yang ditesting memiliki hasil yang sesuai atau tidak.

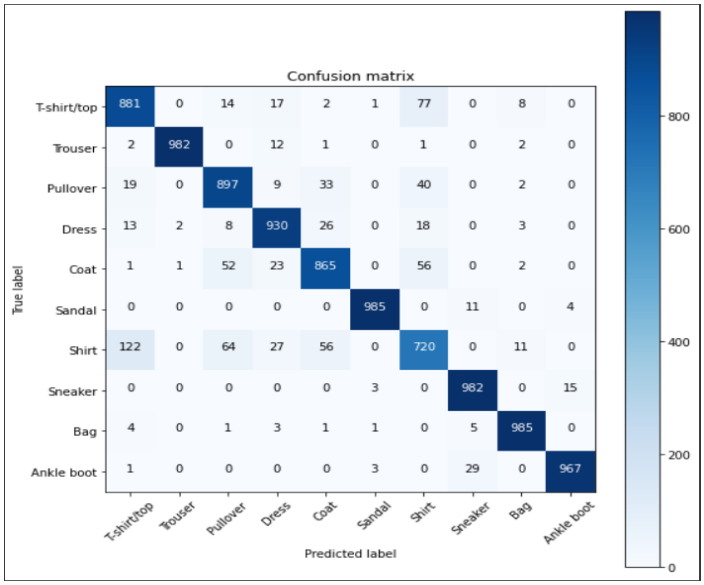
```
score = cnn_model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1]*100)
```

Setiap selesai menghitung mengkalkulasi model tersebut, maka kami akan mencetak accuracy dan loss dari model yang dikalkulasi.

7. Evaluation

```
    def plot_confusion_matrix(cm, classes,title='Confusion matrix',cmap=plt.cm.Blues):
        plt.figure(figsize = (8,8))
        plt.imshow(cm, interpolation='nearest', cmap=cmap)
        plt.title(title)
        plt.colorbar()
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes, rotation=45)
        plt.yticks(tick_marks, classes)

        fmt = 'd'
        thresh = cm.max() / 2.
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, format(cm[i, j], fmt),
                    horizontalalignment="center",
                    color="white" if cm[i, j] > thresh else "black")


        plt.tight_layout()
        plt.ylabel('True label')
        plt.xlabel('Predicted label')
        plt.show()

Y_test_converted = np.argmax(y_test, axis=1)
cnf_matrix = confusion_matrix(Y_test_converted, y_pred)
plot_confusion_matrix(cnf_matrix, classes = class_names)

print(classification_report(Y_test_converted, y_pred))
```

Setelah mendapat hasil evaluasi dari CNN 1 layer, CNN 2 layer, dan CNN 3 layer. Selanjutnya adalah kita akan membuat confusion matrix untuk mengetahui data hasil prediksi yang dibuat oleh model.



```
[31] print(classification_report(Y_test_converted, y_pred))
```

	precision	recall	f1-score	support
0	0.84	0.88	0.86	1000
1	1.00	0.98	0.99	1000
2	0.87	0.90	0.88	1000
3	0.91	0.93	0.92	1000
4	0.88	0.86	0.87	1000
5	0.99	0.98	0.99	1000
6	0.79	0.72	0.75	1000
7	0.96	0.98	0.97	1000
8	0.97	0.98	0.98	1000
9	0.98	0.97	0.97	1000
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

8. Conclusion

Terlihat dari evaluasi diatas, model dapat mengklasifikasi trouser dengan akurasi yang cukup baik namun mengklasifikasikan shirt dengan akurasi yang buruk. Gambar Shirt paling sering mendapat misklasifikasi ke kelas T-Shirt/ top, lalu misklasifikasi shirt ke pullover, dan coat juga banyak. Hal ini membuktikan bahwa model masih belum bisa membedakan dengan terlalu baik antara kelas shirt, pullover, dan coat. Oleh sebab itu dibutuhkan data shirt lebih banyak untuk meningkatkan akurasi prediksi class Shirt. T-shirt juga sering di misklasifikasi sebagai Shirt, ini artinya model kurang mengerti apa itu T-Shirt. Jadi untuk meningkatkan akurasi, kita bisa menambahkan data shirt, T-Shirt, Coat, dan pullover agar modelnya bisa belajar apa itu perbedaannya. Berikut adalah akurasi dari setiap model CNN dengan layer 1,2 dan 3

CNN 1-layer	91.51999950408936
CNN 2-layer	91.43999814987183
CNN 3-layer	91.93999767303467

Daftar Pustaka

[1] Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics*, *50*(9), 3840–3854. https://doi.org/10.1109/tcyb.2020.2983860

[2] Xin, M., & Wang, Y. (2019). Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, *2019*(1). https://doi.org/10.1186/s13640-019-0417-8

[3] Qin, J., Pan, W., Xiang, X., Tan, Y., & Hou, G. (2020). A biological image classification method based on improved CNN. Ecological Informatics, 101093. doi:10.1016/j.ecoinf.2020.101093

[4] Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014). Medical image classification with convolutional neural network. *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. https://doi.org/10.1109/icarcv.2014.7064414

[5] Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016). Breast cancer histopathological image classification using Convolutional Neural Networks. *2016 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2016.7727519

[6] Seo, Y., & Shin, K. (2019). Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, *116*, 328–339. https://doi.org/10.1016/j.eswa.2018.09.022

[7] Zhang, W., Tang, P., & Zhao, L. (2019). *Remote Sensing Image Scene Classification Using CNN-CapsNet. Remote Sensing, 11(5), 494.* doi:10.3390/rs11050494

[8] Xin, M., & Wang, Y. (2019). Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, *2019*(1). https://doi.org/10.1186/s13640-019-0417-8

[9] Geitgey, A. (2016, June 13). Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks. Retrieved June 27, 2021, from Medium website: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721