BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Tahap - tahap yang dilakukan pada penelitian ini dimulai dari pengumpulan data, pengolahan data dan perancangan sistem, pelatihan model, evaluasi model, sampai dengan *deployment* model ke *mobile apps*.

4.1 Pengumpulan Data

Tahapan awal pada penelitian ini adalah pengumpulan data citra. Data citra yang digunakan merupakan citra uang kertas rupiah bagian depan dan bagian belakang yang terdiri dari 6 kelas yaitu 20.000 Asli, 20.000 Palsu, 50.000 Asli, 50.000 Palsu, 100.000 Asli, dan 100.000 Palsu. *Dataset* tersebut diperoleh dari pengambilan citra menggunakan kamera *smartphone* 12 *mini* dengan bantuan fitur *night mode* dengan jarak ± 22 sentimeter diatas objek. Persebaran data pada tiap kelasnya ditunjukan pada tabel 4.1.

Tabel 4.1 Persebaran Data

Kelas *Dataset*	**Jumlah**
20.000 Asli	100
20.000 Palsu	100
50.000 Asli	100
50.000 Palsu	100
100.000 Asli	100
100.000 Palsu	100
Total	**600**

Berikut contoh citra uang kertas rupiah pada masing-masing kelas.

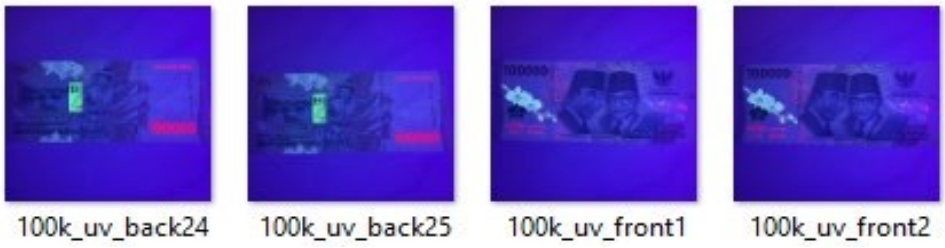
Gambar 4.1 Kelas 20.000 Asli

Gambar 4.2 Kelas 20.000 Palsu

Gambar 4.3 Kelas 50.000 Asli

Gambar 4.4 Kelas 50.000 Palsu

Gambar 4.5 Kelas 100.000 Asli



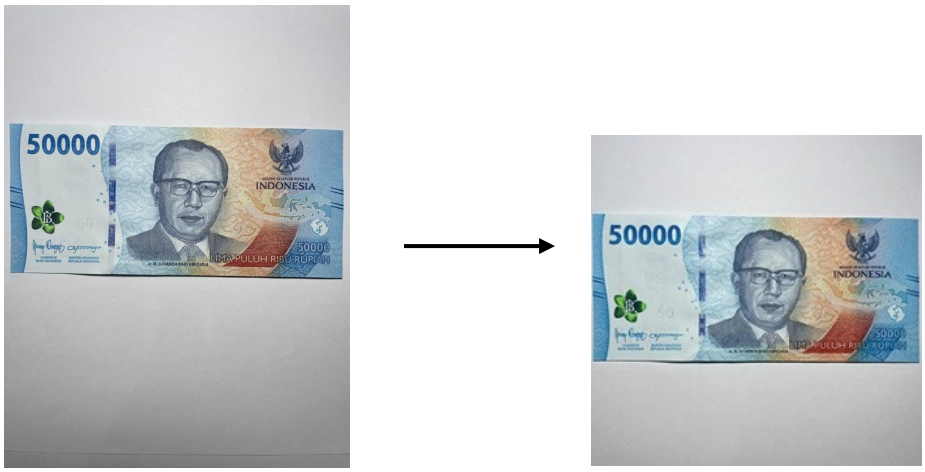
Gambar 4.6 Kelas 100.000 Palsu

4.2 Pengolahan Data dan Perancangan Sistem

Tahapan pengolahan data dan perancangan sistem ini merupakan tahapan dilakukannya pra proses data, perancangan model CNN, dan perancangan model SVM .

4.2.1 Pra Proses Data

Citra yang memiliki resolusi yang besar dan berbeda perlu disesuaikan agar menjadikan proses pengolahan data lebih cepat. Pada penelitian ini, dilakukan proses *cropping* pada ukuran panjang dan lebar citra dengan rasio 1 : 1 dan *resizing* sebesar 299×299 *pixels* menggunakan software Adobe Photoshop CS6. Contoh proses *cropping* dan *resizing* pada citra ditunjukan pada gambar berikut.



Gambar 4.7 Proses *cropping dan resizing* citra uang kertas rupiah

4.2.2 Perancangan Model CNN

Pada penelitian ini proses utama yang harus dilakukan yaitu membuat model klasifikasi menggunakan CNN agar menghasilkan model yang mampu memberikan nilai akurasi yang terbaik. Hal pertama yang dilakukan yaitu dengan memasukkan alamat data dan membaca jumlah data yang telah dibagi setiap kelasnya menggunakan *Google Colaboratory* sebagai media pengolahan pemrograman *python*. Data yang terbaca sudah dibagi menjadi data latih dan data uji dengan rasio 80% : 20%. Data yang terbaca untuk masing – masing enam kelas yaitu 80 citra data latih dan 20 citra data uji.

```
set_1 = os.path.join('/content/drive/My Drive/Uang/train/20.000_asli')
set_2 = os.path.join('/content/drive/My Drive/Uang/train/20.000_palsu')
set_3 = os.path.join('/content/drive/My Drive/Uang/train/50.000-asli')
set_4 = os.path.join('/content/drive/My Drive/Uang/train/50.000_palsu')
set_5 = os.path.join('/content/drive/My Drive/Uang/train/100.000_asli')
set_6 = os.path.join('/content/drive/My Drive/Uang/train/100.000_palsu')
```

```
set_7 = os.path.join('/content/drive/My Drive/Uang/test/20.000_asli')
set_8 = os.path.join('/content/drive/My Drive/Uang/test/20.000_asli')
set_9 = os.path.join('/content/drive/My Drive/Uang/test/50.000_asli')
set_10 = os.path.join('/content/drive/My Drive/Uang/test/50.000_palsu')
set_11 = os.path.join('/content/drive/My Drive/Uang/test/100.000_asli')
set_12 = os.path.join('/content/drive/My Drive/Uang/test/100.000_asli')
```

```
print ('total data train uang 20.000 asli:', len(os.listdir(set_1)))
print ('total data train uang 20.000 palsu:', len(os.listdir(set_2)))
print ('total data train uang 50.000 asli:', len(os.listdir(set_3)))
print ('total data train uang 50.000 palsu:', len(os.listdir(set_4)))
print ('total data train uang 100.000 asli:', len(os.listdir(set_5)))
print ('total data train uang 100.000 palsu:', len(os.listdir(set_6)))
#train test)
print('\n')
print ('total data test uang 20.000 asli:', len(os.listdir(set_7)))
print ('total data test uang 20.000 palsu:', len(os.listdir(set_8)))
print ('total data test uang 50.000 asli:', len(os.listdir(set_9)))
print ('total data test uang 50.000 palsu:', len(os.listdir(set_10)))
print ('total data test uang 100.000 asli:', len(os.listdir(set_11)))
print ('total data test uang 100.000 palsu:', len(os.listdir(set_12)))
```

```
total data train uang 20.000 asli: 80
total data train uang 20.000 palsu: 80
total data train uang 50.000 asli: 80
total data train uang 50.000 palsu: 80
total data train uang 100.000 asli: 80
total data train uang 100.000 palsu: 80


total data test uang 20.000 asli: 20
total data test uang 20.000 palsu: 20
total data test uang 50.000 asli: 20
total data test uang 50.000 palsu: 20
total data test uang 100.000 asli: 20
total data test uang 100.000 palsu: 20
```

Selanjutnya dilakukan proses *rescale* untuk membagi nilai RBG dari 0-255 dengan 255, sehingga didapatkan nilai RGB di rentan 0-1. Lalu, Pada *statement* pemrograman "train_generator" dan "valid_generator" digunakan sebagai proses pembangkitan data berdasarkan citra data latih dan citra data validasi. Kode pemrograman tersebut merubah data berupa *raw image* menjadi dataset yang terdiri dari 480 citra data latih pada 6 kelas dan 120 citra data uji pada 6 kelas untuk digunakan pada proses pembuatan model.

```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale=1./255
                                   )
valid_datagen = ImageDataGenerator(rescale=1./255
                                   )
```

```
train_generator = train_datagen.flow_from_directory('/content/drive/My Drive/Uang/train',
                                                    batch_size = 32,
                                                    target_size=(299,299),
                                                    class_mode="categorical",
                                                    seed=46)
validation_generator = valid_datagen.flow_from_directory('/content/drive/My Drive/Uang/test',
                                                         batch_size = 32,
                                                         target_size=(299,299),
                                                         class_mode="categorical",
                                                         seed=46)
```

```
Found 480 images belonging to 6 classes.
Found 120 images belonging to 6 classes.
```

Untuk mengetahui urutan kelas yang terbaca oleh mesin, digunakan kode pemrograman ”label_map”. Urutan citra uang rupiah yang terbaca yaitu kelas “100.000_Asli” dengan label “0”, kelas“100.000_Palsu” dengan label “1”, “20.000_Asli” dengan label “2” “20.000_Palsu” dengan label “3”, “50.000_Asli” dengan label “4”, “50.000_Palsu” dengan label “5”.

```
label_map=(train_generator.class_indices)
print(label_map)
```

```
{'100.000_asli': 0, '100.000_palsu': 1, '20.000_asli': 2, '20.000_palsu': 3, '50.000-asli': 4, '50.000_palsu': 5}
```

Dalam pembuatan model, digunakan *Library* “Tensorflow” yang merupakan sebuah *framework* yang digunakan untuk melakukan penyelesaian masalah *machine learning* dan *deep learning*. *Library* “Keras” merupakan *library* yang biasa digunakan untuk permasalahan deep learning seperti CNN atau *Recurrent Neural Network* (RNN).

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import regularizers
```

```
model = tf.keras.models.Sequential([
    #konvolusi pertama
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', padding='same', kernel_initializer='he_normal', input_shape=(299, 299, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', padding='same', kernel_initializer='he_normal'), tf.keras.layers.MaxPooling2D(2,2),
    #konvolusi ketiga
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same', kernel_initializer='he_normal'), tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same', kernel_initializer='he_normal'), tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same', kernel_initializer='he_normal'), tf.keras.layers.MaxPooling2D(2,2),
    #flatten dropout
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.3),
    #fully connected layers
    tf.keras.layers.Dense(128, activity_regularizer=regularizers.l2(0.001), activation='relu'),
    tf.keras.layers.Dropout(0.6),
    tf.keras.layers.Dense(6, activation='softmax')
])
```

Dalam Model CNN, arsitektur dapat diatur dan dibuat dengan beberapa layer berbeda beda yang terdiri dari *convolution layer*, *pooling layer*, *flatten layer*, dan *fully-connected layer*. Pada penelitian ini , arsitektur CNN yang dibuat digunakan 3 proses konvolusi dan 3 proses pooling. Konvolusi pertama menggunakan filter sebanyak 16 dengan ukuran kernel sebesar 3x3, fungsi aktivasi yang digunakan dalam konvolusi ini yaitu menggunakan ReLu. Pada penelitian ini digunakan maxpolling dengan ukuran kernel 2x2 mengambil nilai terbesar dari bagian pada area tertentu gambar sehingga menciptakan gambar baru. Pada proses konvolusi kedua menggunakan filter sebanyak 32 karena jumlah masukan pada layer kedua akan semakin kecil sehingga dibutuhkan banyak filter dalam mengekstrak informasi citra. Selanjutnya tetap menggunakan aktivasi ReLu, serta ukuran kernel dan maxpolling yang sama seperti konvolusi pertama. Proses konvolusi ketiga filter ditambah yaitu sebanyak 64, proses maxpolling masih sama seperti konvolusi sebelumnya dengan kernel polling sebesar 2x2, fungsi aktivasi tetap menggunakan ReLu.

Selanjutnya dilakukan *flatten* dan *dropout*. *Flatten* layer terhadap feature maps mengubah nilai masukan menjadi sebuah *array* hasil *polling* yang dilakukan untuk proses *Fully-connected* layer menghasilkan klasifikasi dari citra. *Dropout* Layer digunakan untuk mencegah terjadinya *overfitting* sehingga dapat mempercepat proses learning, pada *dropout* neuron dipilih secara acak dan tidak digunakan selama proses pelatihan, menghilangkan suatu neuron dan menghilangkan sementara jaringan yang ada. *Dense* layer berfungsi untuk menambahkan layer pada *fully-connected* layer. Terdapat *regularizer* yang berfungsi untuk membantu dalam menghindari eror selama proses pelatihan model dan fungsi aktivasi *softmax classifier* sebagai proses klasifikasi karena memberikan hasil yang lebih baik serta lebih perseptif yang membantu dalam proses klasifikasi dalam banyak kelas.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 299, 299, 16)      448

 max_pooling2d (MaxPooling2D  (None, 149, 149, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 149, 149, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 74, 74, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 74, 74, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 37, 37, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 18, 18, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 18, 18, 64)        36928

 max_pooling2d_4 (MaxPooling  (None, 9, 9, 64)         0
 2D)

 flatten (Flatten)           (None, 5184)              0

 dropout (Dropout)           (None, 5184)              0

 dense (Dense)               (None, 128)               663680

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 761,894
Trainable params: 761,894
Non-trainable params: 0
_________________________________________________________________
```

Dengan menggunakan kode “model.summary()”, maka akan akan menampilkan uraian tersebut dari model CNN yang telah dibuat. Jumlah parameter yang dilatih pada model ini sebanyak 761,894 parameter. Ukuran citra pada setiap layernya semakin berkurang, sebelum terakhir masuk ke *fully-connected* layer adalah 9x9 piksel dan jumlah filter yang digunakan sebanyak 64. Lalu dilakukan *reshape* menghasilkan 5184 neuron yang akan masuk pada *fully-connected* layer. Pada *hidden* layer, jumlah neuron yang digunakan sebanyak 512 neuron, dan pada akhirnya klasifikasi yang dilakukan sebanyak 6 kategori.

```
from tensorflow.keras.optimizers import Adam
adam = Adam(learning_rate = 0.001)

#compile
model.compile(loss = 'categorical_crossentropy', optimizer= tf.optimizers.Adam(), metrics=['accuracy'])
```

```
history=model.fit(
    train_generator, steps_per_epoch=3, epochs=100, validation_data=validation_generator, validation_steps=3
    )
```

Dengan kode seperti diatas, dilakukan proses *compile* model dengan menggunakan *optimizer* Adam, *learning rate* sebesar 0.001, *loss function* menggunakan *categorical crossentropy*, dan *metrics* menggunakan *accuracy*. Kemudian pada proses *training* model menggunakan data training yang ada pada training_generator dan data validation dari validation_generator, dengan epoch sebanyak 100 dan step_per_epoch sebanyak 3 step. Hasil dari proses *training* adalah sebagai berikut.

```
Epoch 1/100
3/3 [==============================] - 22s 8s/step - loss: 2.6999 - accuracy: 0.1771 - val_loss: 1.8002 - val_accuracy: 0.1562
Epoch 100/100
3/3 [==============================] - 17s 6s/step - loss: 0.1497 - accuracy: 1.0000 - val_loss: 0.1561 - val_accuracy: 0.9688
```

Gambar 4.8 Hasil proses training model CNN

4.2.3 Perancangan Model SVM

Pada penelitian ini proses utama yang harus dilakukan yaitu membuat model klasifikasi menggunakan algoritma SVM. Hal pertama yang dilakukan yaitu dengan memasukkan alamat data untuk membaca data citra yang akan digunakan melalui *Google Colaboratory* sebagai media pengolahan pemrograman *python*. ". Terdapat kode pemrograman "classes" sebagai pernyataan urutan kelas citra uang rupiah yaitu kelas "100.000_Asli" dengan label "0", kelas"100.000_Palsu" dengan label "1", "20.000_Asli" dengan label "2" "20.000_Palsu" dengan label "3", "50.000_Asli" dengan label "4", "50.000_Palsu" dengan label "5".

```
import os

path = os.listdir('/content/drive/MyDrive/Data_uang/ready')
classes = {'100.000 Asli':0, '100.000 Palsu':1,'20.000 Asli':2,'20.000 Palsu':3,'50.000 Asli':4,'50.000 Palsu':5,}
```

Kode dibawah ini digunakan menyatakan bahwa setiap kelas yang terbaca telah berhasil di simplikasi agar dapat digunakan pada proses selanjutnya.

```
flat_data_arr=[]
target_arr=[]
#please use datadir='/content' if the files are upload on to google collab
#else mount the drive and give path of the parent-folder containing all category images folders.
datadir='/content/drive/MyDrive/Data_uang/ready'
for i in classes:
  print(f'loading... category : {i}')
  path=os.path.join(datadir,i)
  for img in os.listdir(path):
    img_array=imread(os.path.join(path,img))
    img_resized=resize(img_array,(299,299,3))
    flat_data_arr.append(img_resized.flatten())
    target_arr.append(classes[i])
  print(f'loaded category:{i} successfully')
```

Selanjutnya data diubah menjadi bentuk array dan dinyatakan dalam x dan y. selanjutnya dilakukan proses pemisahan data latih dan data uji dengan rasio 80% data latih dan 20% data uji pada setiap kelasnya.

```
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
df=pd.DataFrame(flat_data)
df['Target']=target
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42,stratify=y)
print('Splitted Successfully')
```

Setelah dibagi data uji dan data latih, dilakukan *feature scaling* menggunakan *minmax scaler*, dimana *minmax scaler* mengubah data dari rentan 0 - 255 menjadi 0 - 1 agar data mengolah data. Kemudian dilakukan metode *dimensionality reduction* menggunakan metode *Principle Component Analysis* (PCA) agar mempermudah proses pembelajaran mesin dalam interpretasi data.

```
x_train = x_train/255
x_test = x_test/255
print(x_train.max(), x_train.min())
print(x_test.max(), x_test.min())
```

```
from sklearn.decomposition import PCA
```

```
print(x_train.shape, x_test.shape)

pca = PCA(.98)

pca_train = x_train
pca_test = x_test
```

Model yang sudah dilakukan *feature selection* dilanjutkan dengan proses training menggunakan algoritma SVM seperti kode berikut.

Hyperparameter pada metode SVM digunakan secara default. Artinya, nilai C sebesar 1 dan kernel yang digunakan adalah rbf. Pernyataan ".fit" digunakan untuk melakukan pelatihan dataset menggunakan model SVM.

```
from sklearn.svm import SVC

sv = SVC()
sv.fit(x_train, y_train)
```

4.3 Evaluasi Sistem

Pada tahapan ini dilakukan Evaluasi terhadap model CNN dan model SVM yang telah dibuat. Lalu, hasil dari model terbaik antara kedua algoritma akan diterapkan pada aplikasi berdasarkan nilai *accuracy* tertinggi.

4.3.1 Evaluasi Model CNN

Setelah melakukan proses training, model yang sudah terlatih pada proses *training* dievaluasi menggunakan *validation data* untuk mendapat nilai akhir *accuracy*. Untuk melakukan evaluasi model terhadap data test set menggunakan kode sebagai berikut :

```
model.evaluate(validation_generator)
```

Kemudian didapatkan hasil output seperti gambar 4.9, terlihat didapatkan hasil evaluasi model dengan nilai *loss* 13.71% dan *accuracy* sebesar 98.33%.

```
4/4 [==============================] - 5s 1s/step - loss: 0.1371 - accuracy: 0.9833
[0.1370515376329422, 0.9833333492279053]
```

Gambar 4.9 Hasil Evaluasi Model

Kode dibawah merupakan kode untuk menampilkan grafik menampilkan akurasi dari model yang telah dipilih, grafik menunjukkan seberapa besar akurasi pada model selama dilatih.

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
```

```
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='training accuracy')
plt.plot(val_acc, label='validation accuracy')
plt.legend(loc='lower right')
plt.ylabel('accuracy')
plt.title('training and validation accuracy')
plt.show()
```

Gambar 4.10 menunjukkan grafik *accuracy* selama proses training dan validation berlangsung. Dilihat dari gambar grafik tersebut, model sudah good fit, tidak mengalami *overfitting* ataupun *underfitting*. Artinya, model dapat berjalan dengan baik tanpa eror yang signifikan.

Gambar 4.10 Grafik Pelatihan Model CNN

Selanjutnya dilakukan prediksi untuk mendapatkan hasil pengujian model dari *confusion matrix* menggunakan validasi data. Pada kode dibawah ini, label yang digunakan merupakan kelas dari dataset yang telah ditentukan sebelumnya yaitu, label 0 untuk kelas “100.000 Asli”. Label 1 untuk kelas “100.000 Palsu”. Label 2 untuk kelas “20.000 Asli”. Label 3 untuk kelas “20.000 Palsu”. Label 4 untuk kelas “50.000 Asli”. Label 5 untuk kelas “50.000 Palsu”.

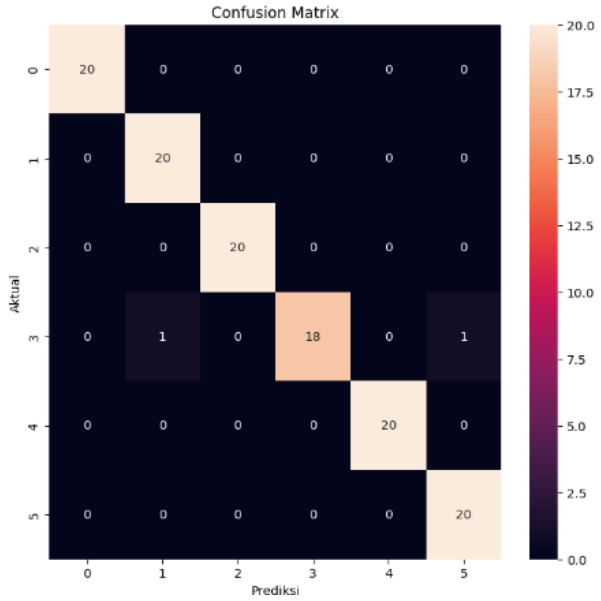
```
import sklearn
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

for x in label_map:
      print(label_map[x],":",x)

prediksi=model.predict_generator(test_data_generator)
y_pred=np.argmax(prediksi,axis=1)

plt.figure(figsize=(8,8))
sns.heatmap(confusion_matrix(test_data_generator.classes,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.ylabel("Aktual")
plt.xlabel("Prediksi")
```

Pada gambar 4.11 menunjukkan bahwa dari total 120 data validasi yang tersebar pada 6 kelas, terdapat 1 kelas yang mengalami eror dalam proses prediksi kelas terhadap aktual kelas.



Gambar 4.11 *Confusion Matrix* pada Model CNN

4.3.2 Evaluasi Model SVM

Pada tahapan evaluasi model, model yang telah berhasil dilakukan training dinyatakan menggunakan kode di bawah ini untuk mendapat hasil score dari proses training yang sudah berlangsung.

```
print("Training Score:", sv.score(x_train, y_train))
print("Testing Score:", sv.score(x_test, y_test))
```

```
Training Score: 0.9729166666666667
Testing Score: 0.9666666666666667
```

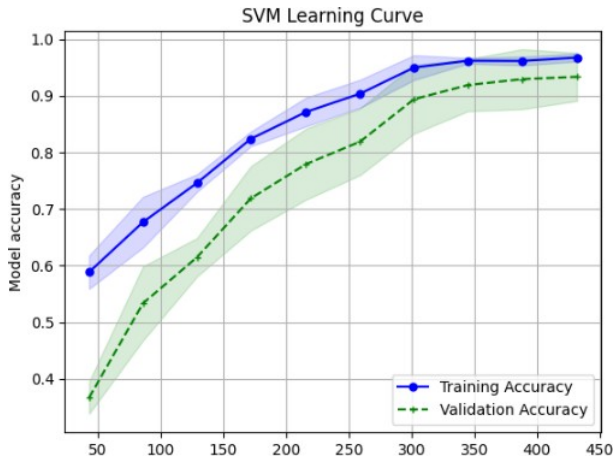
Sesuai dengan *output* pemrograman diatas, nilai *training* yang didapat setelah diubah ke bentuk persen sebesar 97% dan nilai validasi sebesar 96.7%. Lalu, untuk memastikan proses training berjalan dengan baik dan tidak terdapat eror, maka dibuat sebuah *learning curve*. Model yang baik apabila kurva training dan kurva validasi memiliki jarak yang berdekatan. Kode program dibawah ini digunakan untuk membuat *learning curve*.

```
from sklearn.model_selection import learning_curve
```

```
train_sizes, train_scores, test_scores = learning_curve(estimator=sv, X=x_train, y=y_train,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
                                                       n_jobs=1)
```

```
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--', label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15, color='green')
plt.title('SVM Learning Curve')
plt.xlabel('Number of Subset Samples')
plt.ylabel('Model accuracy')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

Pada gambar 4.12, kurva training dan kurva validasi.memiliki jarak yang semakin lama semakin berdekatan. Artinya, proses pelatihan model berjalan dengan baik dan tidak terjadi *overfitting* maupun *underfitting*.

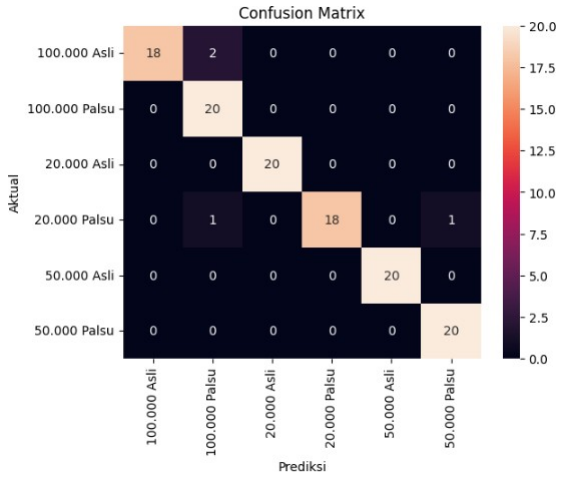


Gambar 4.12 Learning Curve pada model SVM

Kemudian, dilakukan pembuatan *confusion matrix* untuk melihat performa model SVM terhadap validasi data menggunakan kode pemrograman dibawah beserta label kelasnya.

```
cm = confusion_matrix(y_test, y_pred)
sn.heatmap(cm, annot=True, xticklabels=["100.000 Asli","100.000 Palsu", "20.000 Asli", "20.000 Palsu", "50.000 Asli","50.000 Palsu"],
           yticklabels=["100.000 Asli","100.000 Palsu", "20.000 Asli", "20.000 Palsu", "50.000 Asli","50.000 Palsu"])
```

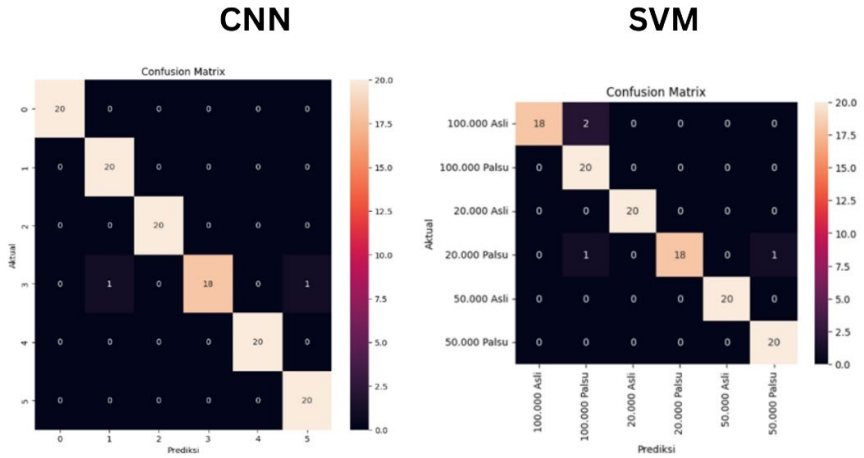
Pada gambar 4.12, menunjukkan bahwa dari total 120 data validasi yang tersebar pada 6 kelas, terdapat 2 kelas yang mengalami eror dalam proses prediksi kelas terhadap aktual kelas.



Gambar 4.13 *confusion matrix* model SVM

4.3.3 Penentuan Model Terbaik

Model dari dua percobaan algoritma CNN dan SVM memiliki performa akurasi yang tidak jauh berbeda. Berdasarkan hasil pelatihan model, masing – masing algoritma mendapatkan nilai *accuracy* sebesar 98.33% untuk CNN dan 96.67% untuk algoritma SVM. Dari Gambar 4.13 yang menunjukkan *confusion matrix* CNN dan SVM menampilkan bahwa terdapat selisih 1 kelas eror dimana aktual kelas dengan label 0 yaitu “100.000 Asli” dapat diprediksi sempurna oleh CNN. Oleh sebab itu nilai *accuracy* model SVM dapat lebih unggul dibandingkan dengan nilai *accuracy* model CNN.



Gambar 4.14 *confusion matrix* model CNN dan SVM

4.4 Penerapan Model pada Aplikasi

Pengimplementasian penelitian ini yaitu pada pembuatan *mobile apps* yang dapat digunakan pada sistem mobile berbasis *android*, model terbaik yang diambil pada penelitian ini yaitu model CNN dengan nilai *accuracy* sebesar 98.33% menggunakan *software* Android Studio.

4.4.1 Tampilan Awal

Tampilan awal ini merupakan pembuka ketika awal membuka aplikasi. Tampilan ini akan muncul beberapa detik lalu melanjutkan ke tampilan berikutnya. Berikut merupakan tampilan awal pada aplikasi.

Gambar 4.15 Tampilan awal pada aplikasi

4.4.2 Tampilan Utama

Tampilan utama merupakan halaman yang terbuka setelah tampilan awal pada aplikasi. Pada tampilan ini digunakan sistem klasifikasi menggunakan model terbaik yaitu model CNN yang telah dibuat sebelumnya. Untuk melakukan unggah gambar dapat menggunakan ambil gambar menggunakan kamera pada *smartphone* atau menggunakan gambar yang terdapat pada *gallery smartphone*. Dibawah ini merupakan tampilan utama yang telah dibuat.

Gambar 4.16 Tampilan utama pada aplikasi

Apabila gambar berhasil diunggah, maka akan menampilkan hasil prediksi dan menampilkan gambar yang diunggah pada aplikasi seperti pada gambar dibawah ini.

Gambar 4.17 Tampilan hasil klasifikasi pada aplikasi

Dalam beberapa kali percobaan seperti contoh pada gambar 4.18, dapat disimpulkan bahwa pendeteksian uang dengan berbagai latar belakang citra yang berbeda dapat membuat sistem kurang maksimal dalam melakukan klasifikasi. Hal ini dapat dikarenakan minim nya variasi dari input dataset maupun minimnya pengembangan pra-pengolahan data yang menyebabkan sistem hanya mengenali data yang berlatar identik.

Gambar 4.18 Hasil Percobaan Klasifikasi pada *Mobile Apps*