DETEKSI PEMALSUAN GAMBAR DENGAN ELA DAN DEEP LEARNING

Agus Gunawan, Holy Lovenia, Adrian Hartarto Pramudita

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

ABSTRAK

Gambar kerap kali dimanipulasi dengan maksud dan tujuan untuk menguntungkan salah satu pihak. Padahal, gambar sering dianggap sebagai bukti dari suatu fakta atau realita, karena itu, berita palsu atau bentuk publikasi apapun yang menggunakan gambar yang sudah dimanipulasi sedemikian rupa memiliki kapabilitas dan potensi untuk menyesatkan yang lebih besar. Untuk mendeteksi pemalsuan gambar tersebut, dibutuhkan data gambar dalam jumlah banyak, dan model yang dapat memproses setiap *pixel* dalam gambar. Selain itu, efisiensi dan fleksibilitas dalam pelatihan data juga dibutuhkan untuk mendukung pemanfaatannya dalam kehidupan sehari-hari. Konsep big data dan deep learning merupakan solusi yang tepat untuk permasalahan ini. Karena itu, dengan arsitektur *Convolutional Neural Network* (CNN) yang memanfaatkan *Error Level Analysis* (ELA), deteksi pemalsuan gambar dapat mencapai 91.83% dan konvergensi hanya dengan 9 *epoch*.

Kata Kunci— *deteksi pemalsuan gambar,* convolutional neural network, error level analysis, deep learning, big data

1. LATAR BELAKANG

Menurut *EU High Level Expert Group* (2018), berita palsu didefinisikan sebagai disinformasi, yaitu segala bentuk informasi yang tidak akurat, salah, ataupun menyesatkan yang dipresentasikan, dipromosikan, atau didesain. Di balik berita-berita palsu, terdapat beberapa alasan mengenai publikasi tersebut. Salah satunya adalah untuk mendapatkan keuntungan secara ekonomi, entah itu melalui peningkatan jumlah klik berita maupun membuat berita yang tidak seharusnya untuk menguntungkan salah satu pihak [1].

Selain itu, berita palsu juga dapat memengaruhi harga saham, yang dapat memberikan keuntungan pada pihak yang merilis berita tersebut. Alasan lainnya adalah untuk mendapatkan dukungan atau menjatuhkan pihak lain secara sosial ataupun politik [2].

Berdasarkan statistik milik Masyarakat Telematika Indonesia (MASTEL) pada tahun 2017, jenis berita palsu yang paling sering diterima adalah sosial-politik, SARA (suku, agama, dan ras), kesehatan, makanan dan minuman, penipuan keuangan, dan IPTEK. Sebanyak 84,5% dari seluruh responden menyatakan bahwa mereka merasa terganggu dengan adanya berita palsu, dan lebih dari 70% sepakat bahwa berita palsu mengganggu kerukunan masyarakat dan menghambat pembangunan.

Selain dalam bentuk tulisan, sekitar 40% responden menyatakan bahwa penyebaran berita palsu juga kerap disertai dengan gambar. Gambar digunakan oleh manusia untuk mereproduksi realita, dan seringkali digunakan sebagai bukti dari suatu berita, publikasi, ataupun fakta. Berita palsu yang memiliki gambar yang mendukung, cenderung diterima dan dipercaya publik.

Secara umum, manusia lebih mudah dalam mengingat bentuk gambar daripada tulisan. Menurut *Social Science Research Network*, sebanyak 65% manusia merupakan orang yang senang belajar melalui visual. Dalam ilmu *marketing* dan *visual*, disebutkan bahwa gambar berpengaruh sangat besar terhadap suatu artikel. Orang-orang cenderung akan memberikan respon ketika ada gambar dibandingkan hanya tulisan. Menurut infografis yang bertemakan tentang pengaruh gambar dalam dunia marketing, gambar dapat menaikkan jumlah responden untuk suatu artikel hingga 94% [8]. Maka dari itu, suatu gambar merupakan elemen yang kuat dalam menyebarkan suatu informasi.

Untuk menentukan suatu gambar asli atau palsu, sangat sulit dilihat dengan mata telanjang, diperlukan teknik-teknik khusus dan ketelitian tertentu agar dapat mengetahui dengan pasti suatu gambar merupakan gambar asli atau sudah mengalami modifikasi. Bagi orang awam, hal ini mungkin sulit untuk dilakukan. Untuk itu, teknologi deteksi pemalsuan gambar ini perlu dikembangkan, agar dapat dimanfaatkan sebagai sarana untuk membantu orang-orang dalam menentukan keaslian suatu gambar.

Teknologi ini membutuhkan banyak data gambar, dan setiap gambar memiliki banyak *pixel-pixel* penyusunnya. Dengan pembelajaran mesin biasa, teknologi ini akan sulit dikembangkan. Sehingga, *big data* dan *deep learning*

merupakan solusi yang tepat untuk menyelesaikan persoalan deteksi pemalsuan gambar ini.

2. TUJUAN

Penambangan data berbentuk deteksi pemalsuan gambar ini memiliki dua tujuan utama sebagai berikut.

1. Mengajukan metode baru dengan menggunakan *deep learning* untuk mengklasifikasi gambar sebagai gambar asli dan gambar yang sudah mengalami modifikasi dengan arsitektur yang lebih sederhana, sehingga biaya komputasi dapat dikurangi
2. Melibatkan penggunaan ELA dalam pembelajaran mesin sebagai usaha untuk meningkatkan efisiensi

Ada beberapa dorongan di balik kedua tujuan utama tersebut. Seperti yang sudah umum diketahui, terdapat beberapa riset lalu yang juga bertujuan untuk mendeteksi pemalsuan gambar [10, 11]. Namun, sebagian besar dari riset-riset tersebut membutuhkan biaya komputasi yang cukup besar (dapat dilihat dari jumlah *epoch* dan *layer* yang dibutuhkan), sehingga fleksibilitas dari metode yang diajukan berkurang dan sulit diaplikasikan pada kehidupan sehari-hari karena terhambat biaya komputasi. Padahal, ada kebutuhan bagi metode deteksi pemalsuan gambar untuk dapat beradaptasi dengan penambahan data gambar asli dan modifikasi seiring berjalannya waktu.

Karena itu, pada makalah ini, diusulkan metode deteksi pemalsuan gambar yang relatif lebih efisien dan memiliki peningkatan skalabilitas yang berbanding lurus dengan pertambahan datanya.

3. MANFAAT

Penambangan data berbentuk deteksi pemalsuan gambar ini dapat dimanfaatkan untuk hal-hal sebagai berikut.

1. Meningkatkan kenyamanan dalam mendapatkan informasi yang sesuai dengan fakta
2. Masyarakat mendapatkan pertimbangan dalam menentukan apakah gambar merupakan asli atau palsu

Dengan adanya referensi bagi masyarakat untuk mengetahui apakah suatu gambar merupakan asli atau bukan, tentunya akan mengurangi keresahan yang ada akibat gambar palsu.

4. BATASAN

Terdapat beberapa batasan yang berlaku pada penambangan data deteksi pemalsuan gambar ini, yaitu data mentahnya harus berupa gambar dengan *lossy compression* (contohnya .jpg), juga bukan merupakan *computer generated image* (CGI).

5. METODE

Terdapat dua metode utama yang digunakan dalam penambangan data ini, yaitu Error Level Analysis (ELA) dan pembelajaran mesin dengan teknik deep learning berupa Convolutional Neural Network (CNN).

5.1. *Error Level Analysis (ELA)*

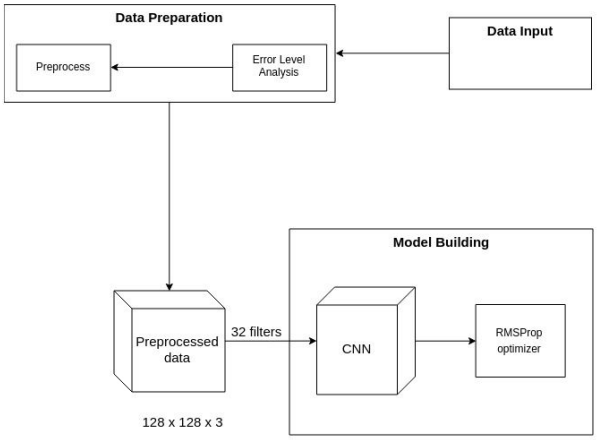
Error Level Analysis merupakan salah satu teknik yang digunakan untuk mendeteksi manipulasi gambar dengan cara menyimpan ulang gambar pada tingkat kualitas tertentu dan menghitung perbandingan antara tingkat kompresinya [4]. Pada umumnya, teknik ini dilakukan pada gambar yang memiliki format *lossy* (*lossy compression*). Tipe gambar yang dipakai dalam penambangan data ini adalah JPEG. Pada gambar JPEG, kompresi dilakukan secara independen untuk setiap 8x8 pixel pada gambar. Jika suatu gambar tidak dimanipulasi, setiap 8x8 pixel pada gambar pasti memiliki tingkat *error* yang sama [6].

5.2. *Convolutional Neural Network (CNN)*

CNN adalah tipe *network* yang berbasis *feedforward*, yang alur informasinya hanyalah satu arah, yaitu dari masukan ke keluaran. Walaupun ada beberapa jenis arsitektur CNN, pada umumnya, CNN memiliki beberapa *convolutional layer* dan *pooling layer*. Kemudian, diikuti oleh satu atau lebih *fully connected layer*. Pada klasifikasi gambar, masukan pada CNN adalah dalam bentuk gambar, sehingga setiap *pixel*-nya dapat diolah [5].

Secara singkat, *convolutional layer* digunakan sebagai pengekstrasi fitur yang mempelajari representasi fitur tersebut dari gambar yang menjadi masukan pada CNN. Sedangkan, pooling layer bertugas untuk mengurangi resolusi spasial dari peta-peta fitur. Umumnya, sebelum *fully connected layer*, terdapat tumpukan beberapa *convolutional* dan *pooling layer* yang berfungsi untuk mengekstrak representasi fitur yang lebih abstrak. Setelahnya, *fully connected layer* akan menginterpretasi fitur-fitur tersebut dan melakukan fungsi-fungsi yang membutuhkan *high-level reasoning*. Klasifikasi pada akhir CNN akan menggunakan fungsi *softmax* [5].

6. DESAIN DAN IMPLEMENTASI



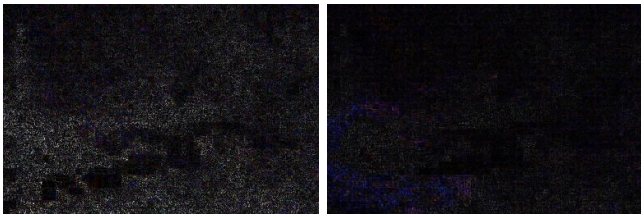
Gambar 1. Arsitektur CNN secara garis besar

Secara umum, desain arsitektur terbagi menjadi dua bagian besar, yaitu *data preparation* dan *model building*. Pada tahap awal, data masukan yang terdiri dari gambar dengan format ".jpg", dengan rincian sebagai berikut: 1771 gambar dengan label *tampered* dan 2940 gambar dengan label *real* [3], dimasukkan ke tahap *data preparation*. Tahap *data preparation* merupakan tahap di mana setiap gambar yang merupakan data masukan dikonversi terlebih dahulu menjadi gambar hasil *Error Level Analysis*. Kemudian, gambar ELA tersebut akan di-*resize* menjadi gambar dengan ukuran 128 x 128.

a) b)

Gambar 2. a) Contoh gambar asli dari kadal dan b) contoh gambar yang sudah dimodifikasi

Konversi data mentah ke gambar hasil ELA merupakan cara yang digunakan untuk menambah efisiensi pelatihan dari model CNN. Efisiensi ini dapat tercapai karena hasil gambar ELA mengandung informasi yang tidak berlebihan seperti gambar aslinya. Fitur yang dihasilkan oleh gambar ELA sudah difokuskan pada bagian gambar yang memiliki level *error* di atas batas. Selain itu, *pixel-pixel* pada gambar ELA cenderung memiliki warna yang mirip atau justru sangat kontras dengan *pixel-pixel* di dekatnya, sehingga pelatihan model CNN menjadi lebih efisien.

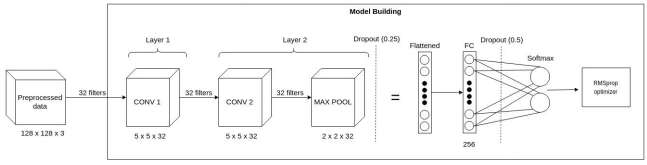


a) b)

Gambar 3. a) Hasil gambar ELA dari Gambar 2a) dan b) hasil gambar ELA dari Gambar 2b)

Setelahnya, dilakukan perubahan ukuran gambar. Langkah selanjutnya adalah melakukan normalisasi dengan membagi setiap nilai RGB dengan angka 255.0 untuk melakukan normalisasi, agar CNN lebih cepat konvergen (mencapai global minimum dari nilai *loss* milik data validasi) karena nilai dari setiap nilai RGB hanya berkisar antara 0 dan 1. Langkah selanjutnya adalah dengan mengubah label pada suatu data, di mana 1 merepresentasikan *tampered* dan 0 merepresentasikan *real* menjadi *categorical value*. Setelah itu dilakukan pembagian data latih dan data validasi menggunakan pembagian 80% untuk data latih dan 20% untuk data validasi.

Langkah selanjutnya adalah menggunakan data latih dan data validasi untuk melakukan pelatihan model *deep learning* dengan menggunakan CNN. Optimasi diterapkan selama pelatihan adalah *RMSProp optimizer*, yang merupakan salah satu metode *adaptive learning rate*. Arsitektur lengkap yang digunakan pada bagian *model building* dapat dilihat pada gambar di bawah atau dengan menggunakan tautan[] yang merupakan gambar arsitektur secara lengkap.



Gambar 4. Arsitektur pembangunan model CNN

Pada model *deep learning* yang digunakan layer pertama CNN terdiri dari *convolutional layer* dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32. Layer kedua CNN terdiri dari *convolutional layer* dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32, dan *Max Pooling layer* dengan ukuran 2x2. Kedua *convolutional layer* yang digunakan menggunakan *kernel initializer glorot uniform*, dan fungsi aktivasi ReLU untuk membuat neuron yang ada pada *convolutional layer* melakukan seleksi sehingga dapat menerima sinyal yang berguna dari data masukan [9].

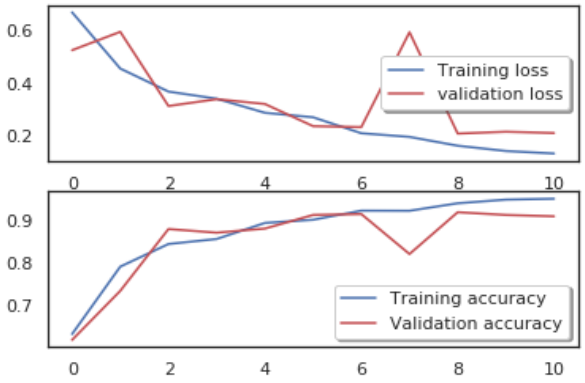
Setelahnya, layer *MaxPooling* ditambahkan *dropout* sebesar 0.25 untuk mencegah *overfitting*. Layer berikutnya

merupakan *fully connected layer* dengan jumlah *neuron* sebanyak 256 dan fungsi aktivasi ReLU. Setelah *fully connected layer*, akan ditambah *dropout* sebesar 0.5 untuk mencegah *overfitting*. Layer *output* yang digunakan memiliki fungsi aktivasi *softmax*.

Pada arsitektur yang digunakan, hanya dua *convolutional layer* yang dibutuhkan, karena hasil yang dihasilkan dari proses konversi menjadi gambar ELA dapat menonjolkan fitur-fitur penting untuk mengetahui apakah sebuah gambar asli atau sudah mengalami modifikasi dengan baik.

7. ANALISIS

Hasil yang didapatkan dari metode yang diajukan memiliki akurasi maksimum sebesar 91.83%. Gambar kurva akurasi dan kurva *loss* dapat dilihat pada gambar di bawah.



Gambar 5. Kurva akurasi dan kurva loss untuk data pelatihan dan data validasi

Dapat dilihat pada gambar di atas bahwa akurasi terbaik didapatkan pada *epoch* ke-9. Nilai *validation loss* setelah *epoch* ke-9 mulai datar dan akhirnya meningkat, yang merupakan tanda dari *overfitting*. Metode identifikasi jumlah *epoch* yang baik untuk digunakan pada saat pelatihan adalah *early stopping*. Dengan metode ini, pelatihan akan dihentikan ketika nilai akurasi validasi mulai menurun atau nilai *validation loss* mulai meningkat.

Jumlah *epoch* pelatihan yang dibutuhkan sedikit untuk mencapai konvergen, karena penggunaan fitur gambar hasil konversi ELA membuat pelatihan model menjadi jauh lebih efisien, dan normalisasi yang dilakukan pada nilai RGB untuk setiap *pixel* juga mempercepat konvergensi dari model CNN.

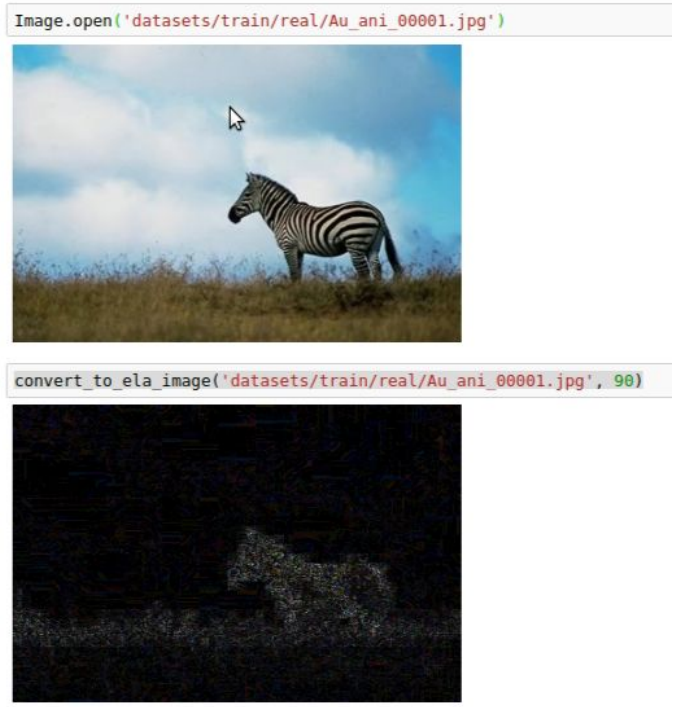
Hasil akurasi yang diperoleh oleh model dalam melakukan klasifikasi dapat dikatakan tergolong tinggi. Hal ini merupakan indikasi bahwa fitur berupa gambar ELA berhasil digunakan untuk melakukan klasifikasi apakah gambar merupakan gambar asli atau sudah mengalami modifikasi.

8. KESIMPULAN

Dalam penelitian ini, terdapat beberapa hal yang dapat disimpulkan dari hasil pembelajaran mesin menggunakan ELA dan CNN.

1. CNN menggunakan dua *convolutional layer*, satu *MaxPooling layer*, satu *fully connected layer*, dan satu *output layer* dengan *softmax* dapat mencapai akurasi 91.83%.
2. Penggunaan ELA dapat meningkatkan efisiensi dan mengurangi biaya komputasi dari proses pelatihan. Hal ini dapat dilihat dari pengurangan jumlah layer dari metode sebelumnya [11] dan jumlah *epoch* yang dibutuhkan. Pada model yang diusulkan, jumlah *epoch* yang dibutuhkan untuk mencapai konvergensi hanyalah 9.

9. DOKUMENTASI



Gambar 6. Konversi gambar asli menjadi gambar hasil ELA

```
Image.open('datasets/train/fake/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg')
```

```
convert_to_ela_image('datasets/train/fake/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg', 90)
```

Gambar 7. Konversi gambar *tampered* ke gambar hasil ELA

```
X_train[0:1]
```

```
array([[[[0.02352941, 0.        , 0.        ],
         [0.01176471, 0.01176471, 0.01176471],
         [0.02352941, 0.01176471, 0.        ],
         ...,
         [0.01176471, 0.01176471, 0.01176471],
         [0.01176471, 0.01176471, 0.02352941],
         [0.        , 0.        , 0.        ]],

        [[0.01176471, 0.        , 0.        ],
         [0.01176471, 0.01176471, 0.        ],
         [0.        , 0.        , 0.        ],
         ...,
         [0.01176471, 0.01176471, 0.        ],
         [0.01176471, 0.01176471, 0.        ],
         [0.        , 0.01176471, 0.03529412]],

        [[0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.02352941, 0.01176471, 0.        ],
         ...,
         [0.02352941, 0.        , 0.        ],
         [0.02352941, 0.01176471, 0.01176471],
         [0.        , 0.        , 0.02352941]],

        ...,

        [[0.02352941, 0.02352941, 0.0627451 ],
         [0.1372549 , 0.01176471, 0.03529412],
         [0.01176471, 0.        , 0.05098039],
         ...,
         [0.24313725, 0.08627451, 0.03529412],
         [0.05098039, 0.05098039, 0.08627451],
         [0.02352941, 0.01176471, 0.0745098 ]],

        [[0.0627451 , 0.0627451 , 0.03529412],
         [0.11372549, 0.08627451, 0.05098039],
         [0.11372549, 0.01176471, 0.03529412],
         ...,
         [0.1372549 , 0.05098039, 0.02352941],
         [0.10196078, 0.10196078, 0.05098039],
         [0.15294118, 0.0745098 , 0.0627451 ]],

        [[0.01176471, 0.05098039, 0.        ],
         [0.02352941, 0.01176471, 0.        ],
         [0.05098039, 0.02352941, 0.03529412],
         ...,
         [0.01176471, 0.01176471, 0.        ],
         [0.03529412, 0.05098039, 0.08627451],
         [0.1254902 , 0.01176471, 0.15294118]]]])
```

Gambar 8. Keluaran data latih dari modul *data preparation*

```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu', input_shape = (128,128,3)))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',
                 activation ='relu'))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(0.25))
print("Input: ", model.input_shape)
print("Output: ", model.output_shape)

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(2, activation = "softmax"))
```

```
Input:  (None, 128, 128, 3)
Output:  (None, 124, 124, 32)
Input:  (None, 128, 128, 3)
Output:  (None, 120, 120, 32)
Input:  (None, 128, 128, 3)
Output:  (None, 60, 60, 32)
```

Gambar 9. Modul *model building*

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 124, 124, 32)      2432
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 120, 120, 32)      25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 60, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 115200)            0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               29491456
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
=================================================================
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0
_________________________________________________________________
```

```
optimizer = RMSprop(lr=0.0005, rho=0.9, epsilon=1e-08, decay=0.0)
```

Gambar 10. Ringkasan dari model

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

early_stopping = EarlyStopping(monitor='val_acc',
                               min_delta=0,
                               patience=2,
                               verbose=0, mode='auto')
```

```
epochs = 30
batch_size = 100
```

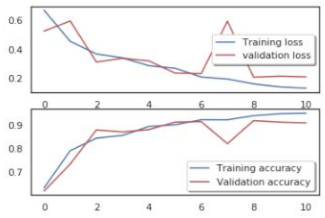
```
history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
          validation_data = (X_val, Y_val), verbose = 2, callbacks=[early_stopping])
```

```
Train on 3768 samples, validate on 943 samples
Epoch 1/30
 - 10s - loss: 0.6697 - acc: 0.6340 - val_loss: 0.5249 - val_acc: 0.6204
Epoch 2/30
 - 7s - loss: 0.4549 - acc: 0.7914 - val_loss: 0.5951 - val_acc: 0.7349
Epoch 3/30
 - 6s - loss: 0.3670 - acc: 0.8442 - val_loss: 0.3119 - val_acc: 0.8791
Epoch 4/30
 - 6s - loss: 0.3398 - acc: 0.8559 - val_loss: 0.3380 - val_acc: 0.8706
Epoch 5/30
 - 6s - loss: 0.2860 - acc: 0.8933 - val_loss: 0.3196 - val_acc: 0.8802
Epoch 6/30
 - 6s - loss: 0.2690 - acc: 0.9007 - val_loss: 0.2350 - val_acc: 0.9120
Epoch 7/30
 - 6s - loss: 0.2080 - acc: 0.9220 - val_loss: 0.2313 - val_acc: 0.9141
Epoch 8/30
 - 6s - loss: 0.1941 - acc: 0.9217 - val_loss: 0.5936 - val_acc: 0.8208
Epoch 9/30
 - 6s - loss: 0.1603 - acc: 0.9392 - val_loss: 0.2067 - val_acc: 0.9183
```

Gambar 11. Pelatihan data

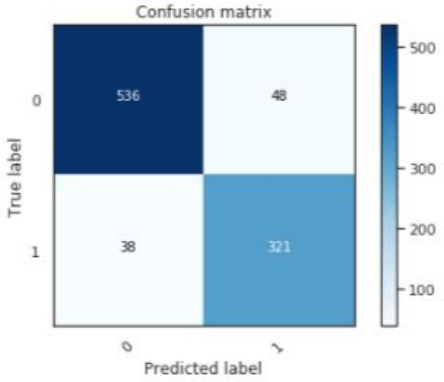
```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```



Gambar 12. Kurva *loss* untuk data latih dan data validasi

```
# Predict the values from the validation dataset
Y_pred = model.predict(X_val)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_val,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2))
```



Gambar 13. *Confusion matrix* dari data validasi (1 melambangkan *tampered*, 0 melambangkan gambar asli)

10. UCAPAN TERIMA KASIH

Ucapan terima kasih penulis untuk penggunaan dataset *CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0* ditujukan kepada *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Corel Image Database* dan para fotografernya.

11. REFERENSI

[1] Özgöbek, Özlem, J. A. Gulla, *"Towards an Understanding of Fake News", Norwegian Big Data Symposium* (2017).

[2] Kshetri, Nir, J. Voas, *"The Economics of 'Fake News'", IT Pro (November/December 2017), IEEE Computer Society*.

[3] *Chinese Academy of Science. "CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0*. Diambil dari http://forensics.idealtest.org

[4] N. Krawetz, *"A pictures worth digital image analysis and forensics," Black Hat Briefings*, hlm. 1-31, 2007.

[5] Rawat, Waseem, Z. Wang, *"Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review"*, *Neural Computation 29* (2017), hlm. 2352-2449.

[6] Gunawan, Teddy Surya, Hanafiah, S. A. M., Kartiwi, M., Ismail, N., Za'bah, N. F., Nordin, A. N., *"Development of Photo Forensics Algorithm by Detecting Photoshop Manipulation Using Error Level Analysis", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 7, No. 1,* Juli 2017, hlm. 131-137.

[6] *Photo Forensics: Detect Photoshop Manipulation with Error Level Analysis*, September 2018. Diambil dari https://resources.infosecinstitute.com/error-level-analysis-detect-image-manipulation/#gref

[7] Edelman, *"2018 Edelman Trust Barometer Global Report"*, diambil dari https://cms.edelman.com/sites/default/files/2018-01/2018%20Edelman%20Trust%20Barometer%20Global%20Report.pdf

[8] Bullas, Jeff. *"6 Powerful Reasons Why you Should include Images in your Marketing"*, diambil dari https://www.jeffbullas.com/6-powerful-reasons-why-you-should-include-images-in-your-marketing-infographic/

[9] V. Nair and G. E. Hinton. *"Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th International Conference on Machine Learning*, 21-24 Juni 2010, hlm. 807-814.

[10] Villan, M. Afsal, Kuruvilla, K., Paul, J., Elias, E. P., *"Fake Image Detection Using Machine Learning", International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 7, No. 2,* 2017.

[11] Rao, Yuan, Ni, J., *"A Deep Learning Approach to Detection of Splicing and Copy-Move Forgeries in Images", 2016 IEEE International Workshop on Information Forensics and Security (WIFS)*.