Program ANN Backpropagation untuk MNIST Handwriting Dataset

Andika Risky Sururi
G6401211046
IPB University
andikarisky@apps.ipb.ac.id

Rafi Fabian Syah
G6401211025
IPB University
rafifbnsyah@apps.ipb.ac.id

Mirza Hafiz M
G6401211017
IPB University
zaamirza@apps.ipb.ac.id

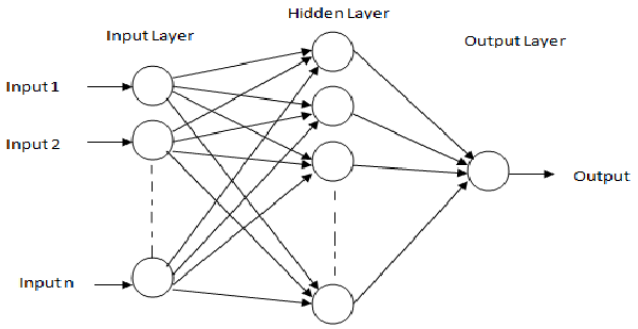
Muhamad Irfan Abdillah
G6401211042
IPB University
irfanabdillah@apps.ipb.ac.id

Stanislaus Brillant Kusuma W.
G6401211063
IPB University
stanislausbrillant@apps.ipb.ac.id

I. PENDAHULUAN

Manusia merupakan makhluk sosial yang bergantung pada interaksi sosial, salah satunya melalui komunikasi. Komunikasi, baik verbal maupun nonverbal, adalah jembatan penting dalam menghubungkan hubungan sosial antar manusia (Kumalasanti 2021). Salah satu bentuk komunikasi nonverbal adalah tulisan tangan, yang menjadi ciri khas dari penulisnya. Namun, tulisan tangan juga sering dimanfaatkan untuk pemalsuan, terutama di tempat kerja atau dalam pendidikan. Oleh karena itu, dibutuhkan sistem yang dapat diandalkan untuk melacak tulisan tangan seseorang berdasarkan kepemilikannya. Pembuatan sistem untuk mengidentifikasi tulisan tangan dapat menggunakan metode klasifikasi.

Menurut Abuzneid dan Mahmood (2018) dalam Maharani *et al.* (2019) *computer vision* memerlukan metode klasifikasi yang kuat untuk mencapai tingkat sistem pengenalan yang tinggi dengan waktu dan sumber daya komputasi yang rendah, salah satunya adalah *Backpropagation Neural Network* (BPNN). BPNN merupakan jaringan syaraf tiruan yang termasuk *supervised learning* dengan *multilayer perceptron* yang memiliki dua arah yaitu: maju dan mundur. BPNN memiliki tiga layer dalam proses pelatihannya yaitu *input layer, hidden layer,* dan *output layer.*



Gambar 1. Arsitektur *Backpropagation Neural Network*

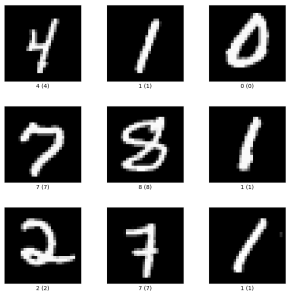
Hidden layer dalam BPNN dapat memperbaiki dan menyesuaikan bobot, serta mengurangi tingkat kesalahan dalam pelatihan jaringan saraf. Hal ini memungkinkan penyesuaian bobot yang lebih baik, sehingga jaringan saraf dapat mendekati target yang diinginkan dengan lebih akurat. Karena keunggulan seperti kesederhanaan, efisiensi, dan kemudahan implementasinya, BPNN sering dipilih untuk tugas pengenalan tulisan tangan. Penelitian ini akan menggunakan teknik BPNN dalam membuat sistem untuk mengembangkan sistem klasifikasi tulisan tangan yang terdiri dari tulisan angka 0 hingga 9 berdasarkan MNIST *Handwritting Dataset*.

II. METODE

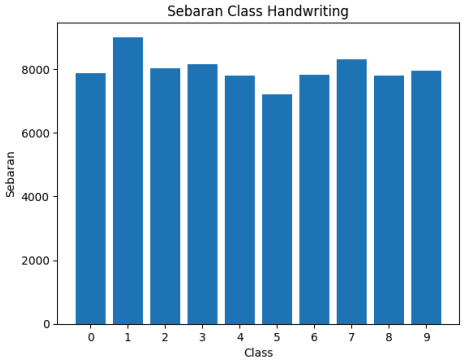
2.1 Pengumpulan Data

Dataset yang digunakan pada penelitian ini merupakan data citra *handwritten* MNIST yang dimuat dari tensorflow dataset. *Dataset* telah dilakukan *data splitting* dengan komponen data 10.000 data uji dan 60.000 data

latih. Data yang digunakan memiliki 10 buah label yang terdiri dari angka 0 hingga 9. Setiap data citra yang dimuat memiliki ukuran 28x28 piksel dengan satu buah channel.



Gambar 2. Dataset Citra *Mnist Handwritten*



Gambar 3. Sebaran kelas

Gambar 2 menunjukkan bagaimana sebaran dari setiap kelas pada data *handwritten* MNIST. Terlihat bahwa setiap kelas memiliki sebaran yang hampir merata berada di antara rentang 6000 hingga 8000 data.

2.2 Pengolahan Data

Tahapan dalam pengolahan data *handwritten* MNIST sebelum digunakan untuk melakukan pelatihan model sebagai berikut.

1. Memuat *dataset* dari Tensorflow
2. *Splitting data* menjadi data latih (*training set*) dan data uji (*testing set*)
3. Melakukan normalisasi piksel sehingga nilai piksel memiliki rentang nilai antara 0 hingga 1

2.3 Membangun Arsitektur dan Melatih Model

Model terdiri dari beberapa *layer*, yakni *input layer*, *hidden layer*, dan *output layer*. Selain itu ditambahkan *layer* untuk melakukan proses konvolusi dan melakukan *max pooling*. *Layer* konvolusi digunakan untuk melakukan ekstraksi fitur pada data citra yang digunakan dalam pelatihan model, sementara *layer max pooling* ditujukan untuk melakukan pengurangan dimensi sehingga proses pelatihan dapat mengurangi penggunaan memori. Dalam pembangunan model ini, *layer* konvolusi dan *max pooling* dilakukan sebanyak dua kali. Setelah dilakukan proses konvolusi dan *max pooling,* selanjutnya dibuat *input layer* dan *hidden layer* sebanyak 128 neuron, serta *output layer* sebanyak 10 neuron.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

Gambar 4. Arsitektur *Neural Network*

Selanjutnya, model dikompilasi dengan menggunakan beberapa parameter, antara lain *optimizer* 'adam', *loss* '*sparse category cross entropy'* dan matriks '*accuracy*'. Setelah arsitektur model dibangun, tahap selanjutnya adalah melakukan proses pelatihan model yang dilakukan dengan menggunakan data latih sebanyak 10 *epochs*.

2.4 Evaluasi Model

Setelah model telah dikonstruksi dan dilakukan proses pelatihan, langkah selanjutnya adalah evaluasi untuk menilai tingkat akurasi model. Evaluasi dilakukan dengan menggunakan matriks akurasi. Evaluasi model juga menggunakan *confusion matrix* untuk melihat sebaran hasil prediksi model, memungkinkan penilaian yang lebih mendalam terhadap kemampuan model dalam memprediksi suatu kelas tertentu.

III. Hasil dan Pembahasan

3.1 Hasil Arsitektur Model

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_8 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_9 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_9 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_4 (Flatten)	(None, 1600)	0
dense_8 (Dense)	(None, 128)	204928
dense_9 (Dense)	(None, 10)	1290

Total params: 225034 (879.04 KB)
Trainable params: 225034 (879.04 KB)
Non-trainable params: 0 (0.00 Byte)

Gambar 5. *Resume* arsitektur model *neural network*

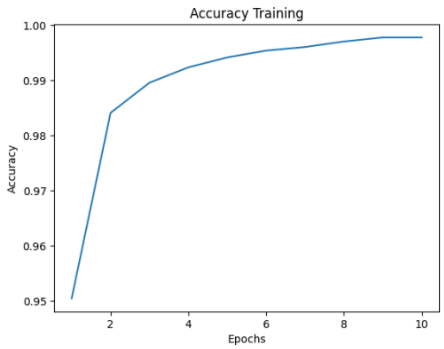
Dalam model yang dibuat, terdapat dua *layer* konvolusi, yakni conv2d_8 dan conv2d_9. Dua *layer* ini berguna untuk menerapkan *filter* pada citra yang diprediksi.

Tujuannya adalah untuk ekstraksi fitur seperti garis dan tepi. Layer konvolusi pertama memiliki 32 filter yang artinya bisa mengidentifikasi 32 fitur yang berbeda dari citra, sedangkan *layer* kedua memiliki 64 filter yang bisa menangkap rentang kompleksitas yang lebih luas.

Melanjutkan dari *layer* konvolusi, terdapat dua *pooling layer*, yakni max_pooling2d dan max_pooling2d_9. *Pooling layer* bertujuan untuk mereduksi dimensi dari data dan mengontrol *overfitting*. Dalam model ini yang digunakan ialah *max pooling* sehingga model mengambil nilai terbesar dalam jendela input untuk setiap *channel* dari citra.

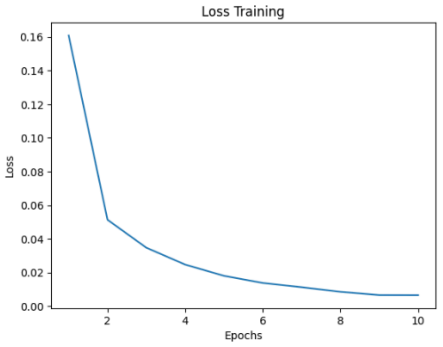
Dari *convolutional layer* dan *pooling layer* dilakukan flatten yang bertujuan untuk mengkonversi data 3 dimensi menjadi vektor satu dimensi. Dalam model ini hasil konversinya memiliki 1600 element yang disimpan dalam flatten_4 *layer*. Dari flatten layer ini dilanjutkan klasifikasinya ke dalam dua layer terakhir yakni dense_8 dan dense_9. *Layer* dense_8 memiliki 128 neuron yang artinya mengklasifikasikan hasil dari flatten *layer* menjadi 128 kelas. Dari dense_8 layer ini dilanjutkan ke dense_9 layer yang memiliki 10 neuron. *Layer* dense_9 mengklasifikasikan hasil yang sebelumnya menjadi 10 kelas digit (0-9).

3.2 Hasil Training dan Evaluasi Model



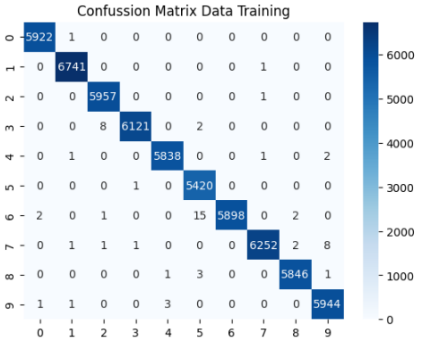
Gambar 6. Pergerakan akurasi *training* setiap *epochs*

Gambar 6 menunjukan visualisasi akurasi pada pelatihan dengan diagram garis, sumbu x pada diagram menunjukan jumlah *epochs* yaitu hingga 10 *epoch*, sumbu y pada diagram menunjukan akurasi model setiap *epoch*, dimana nilai 1,0 menunjukan model yang sempurna dan 0 menunjukan model yang tidak efektif. Berdasarkan hasil tersebut dapat dilihat bahwa garis akurasi cenderung meningkat seiring berjalannya *epoch*, ini menunjukan bahwa model dapat mempelajari data latih dengan baik.



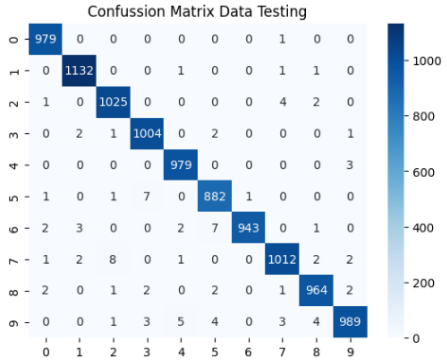
Gambar 7. Pergerakan *loss training* setiap *epochs*

Gambar 7 menunjukan visualisasi *loss* model pada pelatihan dengan diagram garis, sumbu x pada diagram menunjukan jumlah *epochs* yaitu hingga 10 *epoch*, sumbu y pada diagram menunjukan nilai *loss* model setiap *epoch* dimana semakin rendah nilai *loss*, maka semakin baik model yang dihasilkan. Berdasarkan hasil tersebut, dapat dilihat bahwa garis *loss* cenderung menurun seiring berjalannya *epoch*, ini menunjukan bahwa model semakin meningkat kemampuanya untuk membuat prediksi yang akurat.



Gambar 8. *Confusion Matrix* data *training*

Gambar 8 menunjukan hasil *confusion matrix* menggunakan data training. Terlihat bahwa model yang digunakan membentuk hasil korespondensi secara diagonal sehingga dapat dikatakan sudah bisa mengklasifikasikan dataset training ke dalam 10 kelas dengan baik.



Gambar 9. *Confusion Matrix* data *testing*

Gambar 9 menunjukan hasil *confusion matrix* dengan menggunakan data testing. Hasil yang didapat sama seperti ketika menggunakan data training, yakni membentuk diagonal yang artinya digit input dengan output prediksi sudah sesuai untuk 10 kelas prediksi.

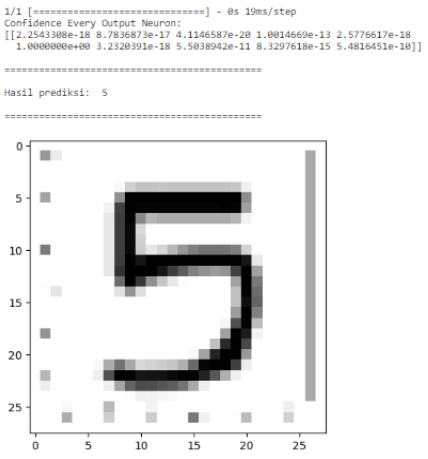
```
loss, accuracy = model.evaluate(x_test,y_test)
print(loss)
print(accuracy)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0371 - accuracy: 0.9909
0.03713436424732208
0.9908999800682068
```

Gambar 10. Akurasi dan *loss* pada data testing

Berdasarkan hasil pengujian model, didapatkan nilai akurasi sebesar 0,99 dan nilai *loss* sebesar 0.03. nilai tersebut menunjukan bahwa model berhasil melakukan prediksi dengan tingkat keakuratan yang tinggi.

3.3 Contoh Hasil Prediksi



Gambar 11. Contoh Hasil Prediksi

Sebagai contoh, hasil prediksi sistem terhadap citra seperti yang ditunjukkan pada gambar 11 menampilkan *confidence every output neuron* dalam bentuk probabilitas setiap kelas. Dapat dilihat bahwa probabilitas pada kelas 5 adalah yang paling besar, menunjukkan bahwa prediksi citra tersebut adalah angka 5.

IV. Kesimpulan

Berdasarkan hasil pengujian model ANN *backpropagation* untuk MNIST *dataset* diperoleh kesimpulan sebagai berikut:

1. Tingkat akurasi dari model adalah 0.9908999800682068 ≈ 0.9909 atau 99.09%
2. Tingkat *loss* dari model adalah 0.03713436424732208 ≈ 0.0371 atau 3.71%
3. Akurasi terbaik pada epoch ke-10
4. Kemampuan model untuk memprediksi *data training* dan *data testing* tidak terlalu berbeda berdasarkan *confusion matrix*, menunjukkan bahwa model tidak mengalami *overfitting*.

Referensi

[1] Kumalasanti RA. 2021. Writer identification based on handwriting using artificial neural network. *International Journal of Applied Sciences and Smart Technologies*. 3(2): 257-264. p-ISSN 2655-8564

[2] Maharani MK, Saputra KO, Wirastuti NMAED. 2022. Komparasi metode backpropagation neural network dan convolutional network pada pengenalan pola tulisan tangan. *J-COSINE (Journal of Computer Science and Informatics Engineering*. 6(1): 56-63. E-ISSN: 2541-0806.