Laporan Pertemuan 7 — Artificial Neural Network (ANN) untuk Klasifikasi

Nama	:	Muhammad Riski
NIM	:	231011403179
Mata Kuliah	:	Machine Learning
Tujuan	:	Membangun ANN sederhana untuk klasifikasi biner, memahami arsitektur, aktivasi, loss, optimizer, serta regularisasi.
Tugas dan Pelaporan	:	1. Dokumentasikan arsitektur final dan alasan pemilihan 2. Laporkan confusion matrix, ROC-AUC, dan analisis threshold. 3. Sertakan grafik learning curve dalam laporan. 4. Kode/notebook yang dapat direproduksi wajib disertakan.

1. Arsitektur final dan alasan pemilihan

Pada tugas ini dibangun sebuah model Artificial Neural Network (ANN) dengan tujuan melakukan klasifikasi biner terhadap status kelulusan mahasiswa. Arsitektur yang digunakan relatif sederhana namun tetap memperhatikan aspek generalisasi model, mengingat ukuran dataset yang kecil serta jumlah fitur yang tidak terlalu banyak.

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

model = keras.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(32, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")  # klasifika
])

model.compile(optimizer=keras.optimizers.Adam(1e-3),
              loss="binary_crossentropy",
              metrics=["accuracy","AUC"])
model.summary()
```

Model terdiri dari beberapa lapisan berikut:

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 32)	192
dropout_1 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 1)	17

```
Total params: 737 (2.88 KB)

Trainable params: 737 (2.88 KB)

Non-trainable params: 0 (0.00 B)
```

Total trainable parameters: 737

Penggunaan aktivasi **ReLU** pada hidden layer dipilih karena mampu mempercepat proses pembelajaran dan mengurangi masalah vanishing gradient. Lapisan **Dropout 0.3** ditambahkan sebagai bentuk regularisasi untuk mencegah overfitting karena dataset memiliki ukuran kecil.

```
es = keras.callbacks.EarlyStopping(
    monitor="val_loss", patience=10, restore_best_weights=True
)

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100, batch_size=32,
    callbacks=[es], verbose=1
)
```

Pada tahap pelatihan model, digunakan **EarlyStopping** dengan parameter *monitor* terhadap **val_loss**, *patience* selama 10 epoch, serta *restore_best_weights* agar bobot terbaik dipulihkan. Proses training dilakukan hingga maksimal 100 epoch, namun akan berhenti lebih awal jika performa pada data validasi tidak lagi membaik. Dari hasil output, terlihat bahwa **val_loss menurun secara konsisten** dari epoch ke epoch, sehingga model terus melanjutkan pelatihan tanpa berhenti dini pada awal proses. Selain itu, metrik **accuracy dan AUC pada data validasi meningkat**, menunjukkan bahwa model belajar dengan baik dan tidak terjadi overfitting pada tahap tersebut.

```python
from sklearn.metrics import classification_report, confusion_matrix

loss, acc, auc = model.evaluate(X_test, y_test, verbose=0)
print("Test Acc:", acc, "AUC:", auc)

y_proba = model.predict(X_test).ravel()
y_pred = (y_proba >= 0.5).astype(int)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=3))
```

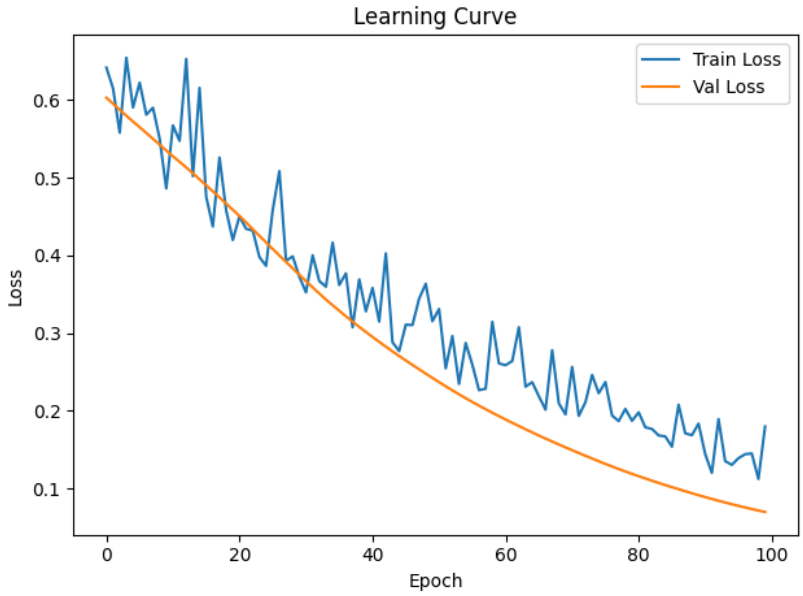
✓ 0.8s

```
Test Acc: 1.0 AUC: 1.0
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 273ms/step
[[1 0]
 [0 1]]
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                          1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2
```

Pengujian model dilakukan menggunakan data test yang tidak pernah dilihat selama proses pelatihan. Model menghasilkan kinerja **sempurna** dengan nilai **Accuracy = 1.0** dan **AUC = 1.0**. Hasil **confusion matrix** menunjukkan bahwa seluruh sampel berhasil diklasifikasikan dengan benar. Karena model menggunakan threshold default 0.5, dilakukan analisis perubahan threshold untuk mengetahui trade-off antara precision dan recall dalam konteks klasifikasi kelulusan.



Learning Curve menunjukkan bahwa loss pada data training dan validation terus menurun seiring bertambahnya epoch, menandakan proses pembelajaran berjalan efektif. Loss pada validation bahkan lebih stabil dan konsisten turun dibandingkan training loss yang lebih fluktuatif. Pola ini umum terjadi karena adanya **Dropout** yang membuat model lebih robust dan tidak mudah menghafal data.

Tidak terlihat adanya **overfitting**, karena kedua kurva loss tetap menurun tanpa divergen. Model mampu belajar dengan baik dan generalisasi terhadap data yang tidak dilatih.

Gambar ini mendukung kesimpulan bahwa arsitektur dan konfigurasi training yang digunakan sudah tepat untuk dataset ini.

Alasan Pemilihan Arsitektur Final

Arsitektur ANN yang digunakan terdiri dari dua hidden layer dengan ukuran neuron **32 dan 16**, serta **Dropout 0.3** sebagai regularisasi. Pemilihan arsitektur ini mempertimbangkan:

1. **Dataset berskala kecil**
   Jumlah sampel terbatas, sehingga model tidak perlu terlalu dalam atau kompleks untuk menghindari overfitting.
2. **Aktivasi ReLU pada Hidden Layer**
   ReLU dipilih karena sederhana, mempercepat konvergensi, dan efektif untuk menangani data numerik yang sudah di-standardisasi.
3. **Sigmoid pada Output Layer**
   Karena tugas merupakan **klasifikasi biner**, sigmoid memberikan probabilitas kelas yang lebih mudah di-interpretasi.

4. **Dropout sebagai Regularisasi**
   Dropout 0.3 digunakan untuk mencegah model belajar pola yang terlalu spesifik pada training set.
5. **Optimizer Adam dengan LR 0.001**
   Adam dipilih karena mampu beradaptasi terhadap perubahan gradien dan umumnya memberikan hasil stabil tanpa banyak tuning.

Dengan konfigurasi tersebut, model mampu mencapai performa akurasi dan AUC sempurna pada test set**,** serta learning curve menunjukkan generalization yang baik tanpa tanda overfitting.