ANN (Artificial Neural Network)

Adinda Putri - 13523071

Artificial Neural Network bekerja dengan cara meniru bagaimana otak manusia memproses informasi. Sesuai dengan namanya, ANN menggunakan artificial neurons untuk menganalisis data, mengidentifikasi pola, dan membuat prediksi. Ide utama dari ANN adalah model ini bisa belajar dari data yang diproses seperti otak manusia yang belajar dari pengalaman. ANN terdiri dari tiga layer utama, yaitu input layer, hidden layers, dan output layer. Penjelasan dari masing-masing layer tersebut adalah sebagai berikut:

- Input Layer: Layer tempat ANN menerima informasi. Contohnya, pada image recognition task, input dapat berupa gambar.
- Hidden Layers: Layer-layer ini memproses data dari input layer. Semakin banyak hidden layers, semakin kompleks pola yang bisa dipelajari dan dipahami ANN. Setiap hidden layer mentransformasi data menjadi informasi yang lebih abstrak
- Output Layer: Layer tempat keputusan atau prediksi akhir dibuat. Contohnya, setelah memproses sebuah gambar, output layer memutuskan bahwa seekor hewan adalah kucing atau anjing.

Cara Kerja

1. Inisialisasi model beserta parameter-parameternya dengan:

- Weight diisi sesuai metode (He, Xavier, atau random)

- Bias b = nol

- Jika Adam (Adaptive Moment Estimation) digunakan, inisialisasi momentum m dan varians v ke nol

2. Forward Propagation: Hitung output layer untuk input X. Untuk tiap layer l:

1. **Pre-activation:**

$$Z^{[l]} = A^{[l-1]}W^{[l]T} + b^{[l]T}$$

2. **Activation:**

$$A^{[l]} = f(Z^{[l]})$$

3. Hitung Loss: Hitung loss dengan **MSE:**

$$L = \frac{1}{2m} \sum_{i=1}^{m} (y_{pred}^{(i)} - y^{(i)})^2$$

atau dengan **Binary Cross-Entropy:**

$$L = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log y_{pred}^{(i)} + (1 - y^{(i)}) \log(1 - y_{pred}^{(i)}) \right]$$

Tambahkan penalti jika ada regularisasi l1/l2.

4. Backward Propagation: Hitung gradient loss terhadap parameter.

1. **Output layer:** Untuk BCE + sigmoid:

$$dZ^{[L]} = A^{[L]} - y$$

   Sedangkan untuk MSE, turunan sesuai aktivasi

2. **Gradient parameter:**

$$dW^{[l]} = \frac{1}{m} dZ^{[l]T} A^{[l-1]}$$

$$db^{[l]} = \frac{1}{m} \sum dZ^{[l]}$$

3. **Backpropagation ke layer sebelumnya:**

$$dA^{[l-1]} = dZ^{[l]} W^{[l]}$$

5. Update Bobot: Gunakan gradien untuk memperbarui parameter.

- Gradient Descent:

$$W^{[l]} := W^{[l]} - \eta dW^{[l]}$$

$$b^{[l]} := b^{[l]} - \eta db^{[l]}$$

- Adam:

Dengan momentum m dan varians v:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) dW$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) dW^2$$

Bias correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Update:

$$W := W - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

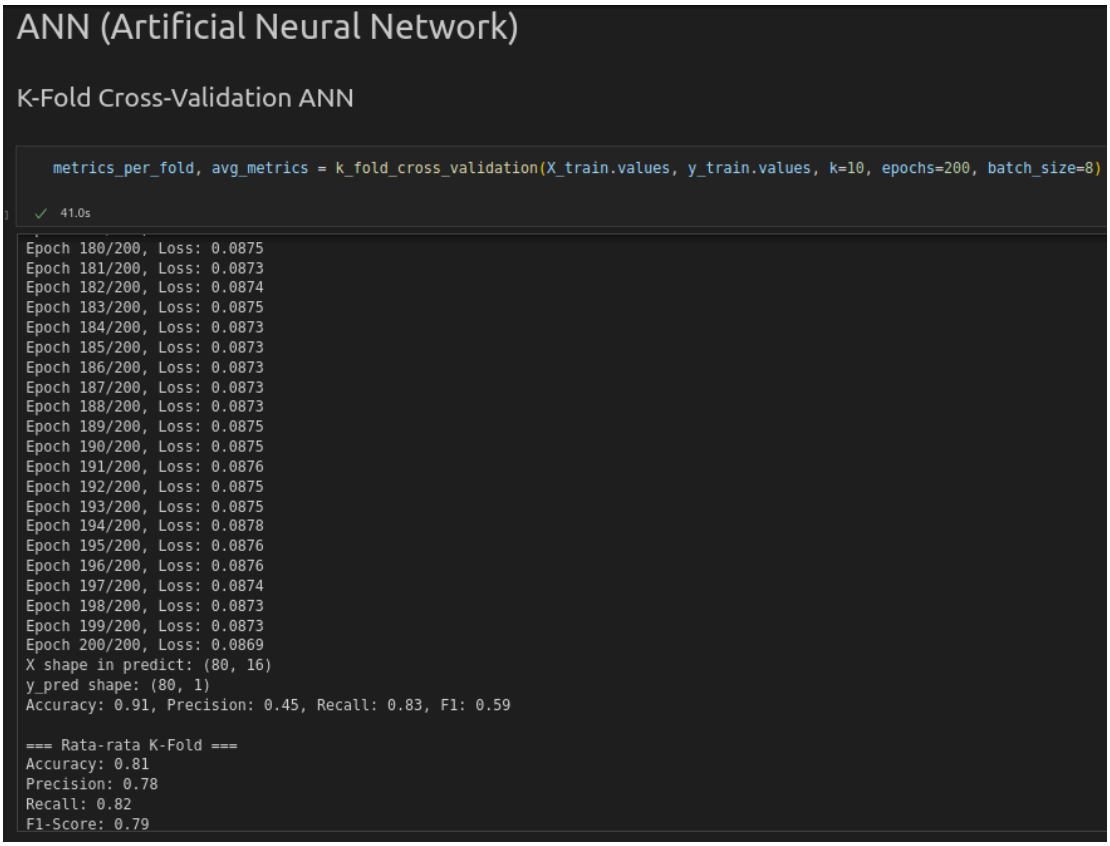
6. Training Loop

Untuk tiap epoch:

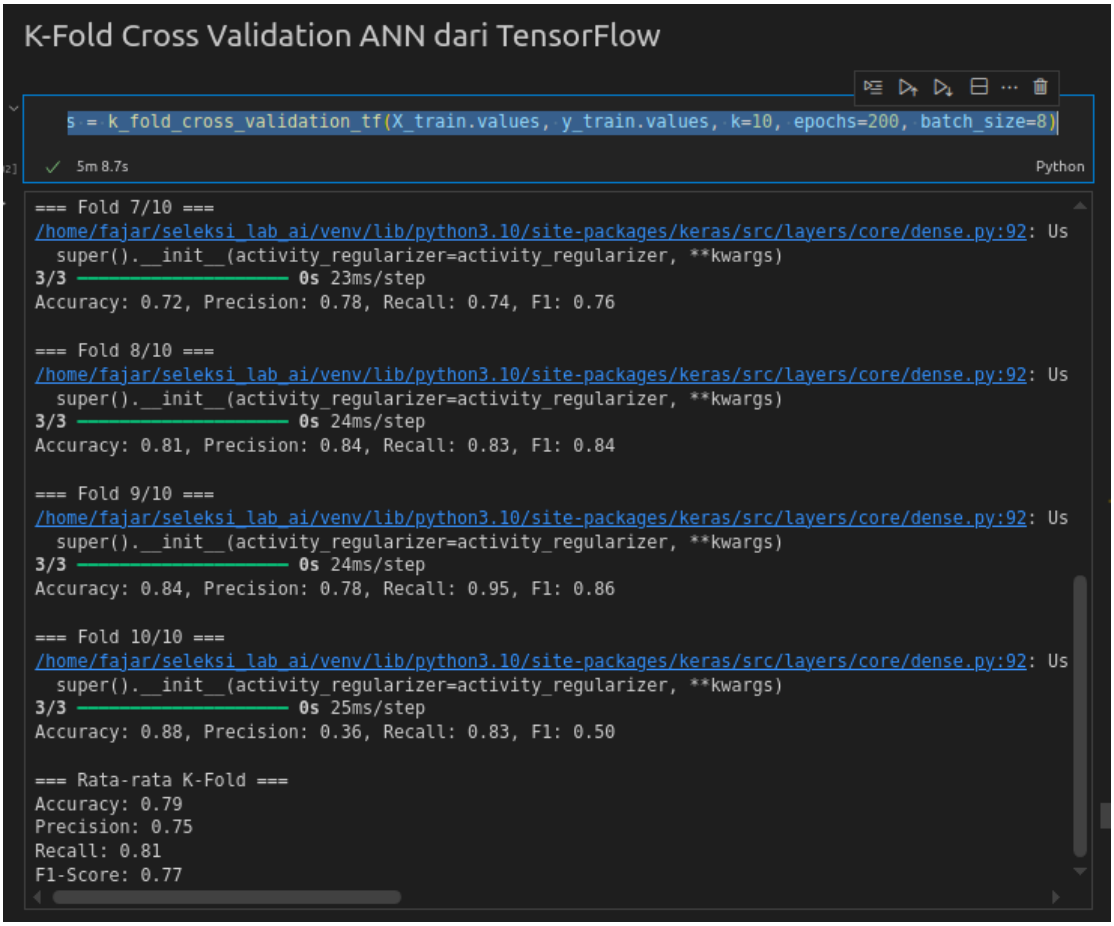
1. Bagi data menjadi batch
2. 2. Untuk tiap batch:
   - Lakukan forward propagation untuk mendapatkan prediksi
   - Hitung loss.
   - Lakukan backpropagation untuk mendapatkan gradient
   - Update bobot
3. Simpan loss tiap epoch

7. Prediction: Untuk input X, lakukan forward propagation untuk mendapatkan output y_pred.

Perbandingan model dari scratch dengan dari TensorFlow



Gambar 1. K-Fold Cross-Validation ANN dari Scratch
Sumber: Penulis



Gambar 2. K-Fold Cross-Validation ANN dari TensorFlow
Sumber: Penulis

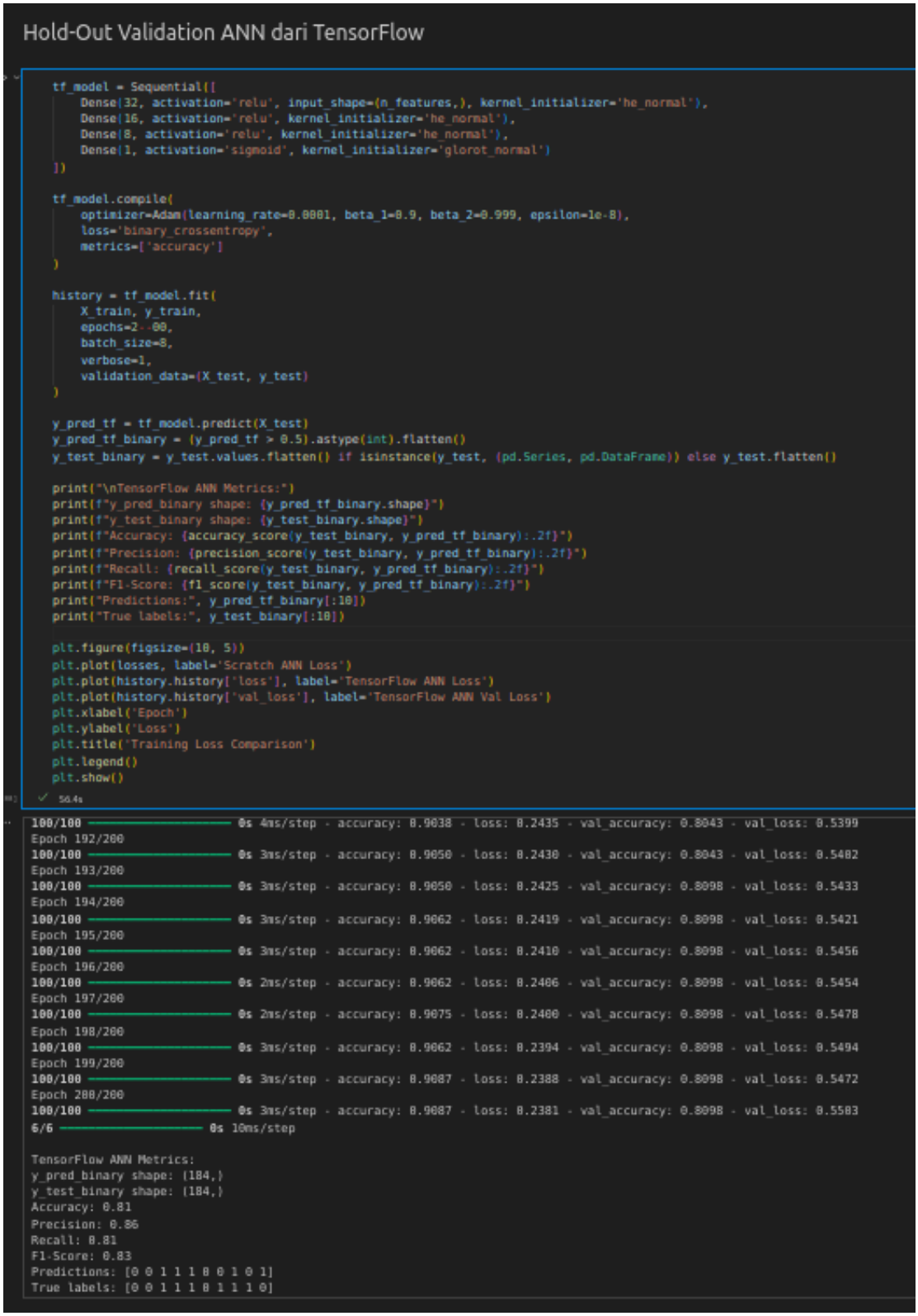
Hold-Out Validation ANN

```
n_features = X_train.shape[1]
ann = ANN(
    layer_sizes=[n_features, 32, 16, 8, 1],
    activations=['relu', 'relu', 'relu', 'sigmoid'],
    init_methods=['he', 'he', 'he', 'xavier'],
    loss='binary_cross_entropy',
    regularization=None,
    lambda_reg=0.0001,
    optimizer='adam',
    lr=0.0001,
    optimizer_params={'beta1': 0.9, 'beta2': 0.999, 'epsilon': 1e-8}
)
losses = ann.fit(X_train, y_train, epochs=200, batch_size=8)

y_pred = ann.predict(X_test)
y_pred_binary = (y_pred> 0.5).astype(int).flatten()
y_test_binary = y_test.values.flatten() if isinstance(y_test, (pd.Series, pd.DataFrame)) else y_test.flatten()

print("ANN Metrics:")
print(f"y_pred_binary shape: {y_pred_binary.shape}")
print(f"y_test_binary shape: {y_test_binary.shape}")
print(f"Accuracy: {accuracy_score(y_test_binary, y_pred_binary):.2f}")
print(f"Precision: {precision_score(y_test_binary, y_pred_binary):.2f}")
print(f"Recall: {recall_score(y_test_binary, y_pred_binary):.2f}")
print(f"F1-Score: {f1_score(y_test_binary, y_pred_binary):.2f}")
print("Predictions:", y_pred_binary[:10])
print("True labels:", y_test_binary[:10])
```

87] ✓ 4.6s

```
Epoch 182/200, Loss: 0.0551
Epoch 183/200, Loss: 0.0546
Epoch 184/200, Loss: 0.0543
Epoch 185/200, Loss: 0.0538
Epoch 186/200, Loss: 0.0536
Epoch 187/200, Loss: 0.0531
Epoch 188/200, Loss: 0.0529
Epoch 189/200, Loss: 0.0522
Epoch 190/200, Loss: 0.0520
Epoch 191/200, Loss: 0.0518
Epoch 192/200, Loss: 0.0516
Epoch 193/200, Loss: 0.0513
Epoch 194/200, Loss: 0.0509
Epoch 195/200, Loss: 0.0507
Epoch 196/200, Loss: 0.0505
Epoch 197/200, Loss: 0.0501
Epoch 198/200, Loss: 0.0499
Epoch 199/200, Loss: 0.0495
Epoch 200/200, Loss: 0.0492
X shape in predict: (184, 16)
y_pred shape: (184, 1)
ANN Metrics:
y_pred_binary shape: (184,)
y_test_binary shape: (184,)
Accuracy: 0.80
Precision: 0.87
Recall: 0.78
F1-Score: 0.82
Predictions: [0 0 1 1 1 0 0 1 1 0]
True labels: [0 0 1 1 1 0 1 1 1 0]
```

Gambar 3. Hold-Out Validation ANN dari Scratch
Sumber: Penulis



Gambar 4. Hold-Out Validation ANN dari TensorFlow
Sumber: Penulis

Berdasarkan hasil evaluasi k-fold cross-validation dan hold-out validation di atas, implementasi ANN dari scratch menunjukkan performa yang baik bahkan sedikit lebih unggul dibanding TensorFlow dalam beberapa metrik, yaitu:

- Accuracy lebih tinggi pada k-fold validation
- F1-Score konsisten lebih baik pada k-fold validation
- Precision pada hold-out validation hampir sama dan pada k-fold validation berdekatan.

Hal ini menunjukkan bahwa ANN dari scratch berhasil diimplementasikan karena konsistensi hasil, stabilitas training dan performa yang kompetitif. Perbedaan kecil pada kedua model dapat terjadi karena perbedaan beberapa parameter untuk optimasi pada tensorflow tidak ada pada model scratch dan presisi numerik

Ruang Improvement

- Masih terdapat beberapa kode belum memanfaatkan secara penuh vectorized operation untuk mempercepat waktu eksekusi.
- Bisa dicoba early stopping berdasarkan validation loss.

Referensi:

[1] *Artificial Neural Networks and its Applications*, GeeksforGeeks. [Daring]. Tersedia: https://www.geeksforgeeks.org/artificial-intelligence/artificial-neural-networks-and-its-applications/. [Diakses: 5 September 2025].

[2] *Deep Learning Course — Lesson 5: Forward and Backward Propagation*, Medium oleh Deep Learning Course. Dipublikasikan 26 Mei 2023. [Daring]. Tersedia: https://medium.com/@nerdjock/deep-learning-course-lesson-5-forward-and-backward-propagation-ec8e4e6a8b92. [Diakses: 5 September 2025].