UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
EST. 1849

Kelompok 4

PROYEK AKHIR PENGOLAHAN SINYAL MEDIS

CLASSIFYING BONE MARROW CELLS WITH CNN

Adonnai Phylosophia
2206060712

Kizza Naufal
2206026504

Haydar Alsagaff
2206060580

UNIVERSITAS INDONESIA
PROGRAM STUDI
TEKNIK BIOMEDIK

AGENDA OVERVIEW

01

MOTIVATION & RELEVANCE

02

DATASET SNAPSHOT AND EDA

03

TRAINING BASELINE VS. ADVANCED RESULTS

04

GUI ARCHITECTURE

05

LIVE/RECORDED GUI DEMO

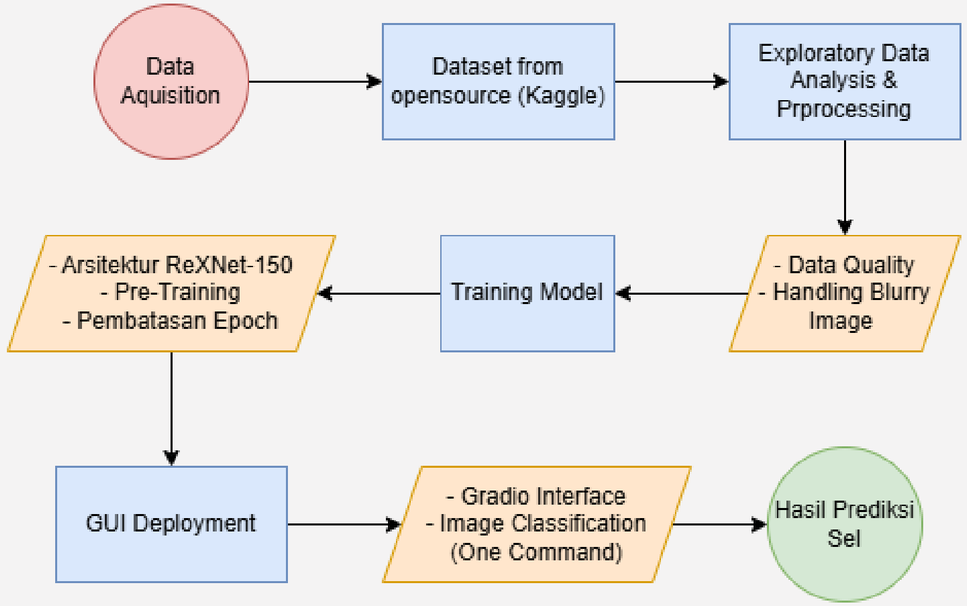
06

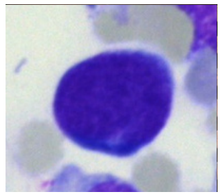
LIMITATIONS, TAKE-HOME MESSAGE, FUTURE WORK

MOTIVATION & RELEVANCE

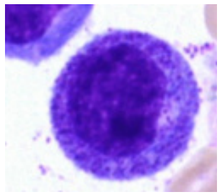
Mengapa Klasifikasi Sel Sumsum Tulang dengan AI Penting?

- Morfologi sumsum tulang adalah fondasi diagnosis hematologi
- Klasifikasi manual memakan waktu dan bergantung pada ahli
- Deep learning memberikan klasifikasi yang akurat, skalabel, dan konsisten
- Berpotensi memperluas akses diagnosis di daerah terbatas sumber daya

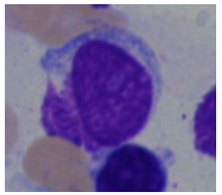




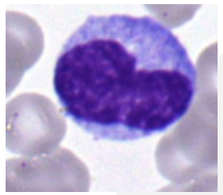
BLA -BLAST
n = 500 I 200



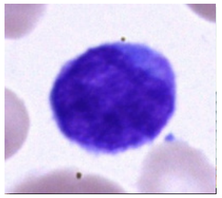
EOS - EOSINOPHIL
n = 500 I 200



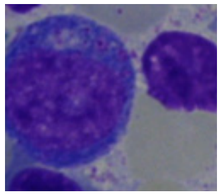
LYT - LYMPHOCYTE
n = 500 I 200



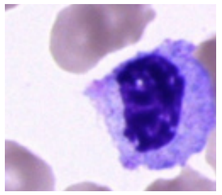
MON - MONOCYTE
n = 500 I 200



NGS - SEGMENTED NEUTROPHIL
n = 500 I 200



PMO - PROMYELOCYTE
n = 500 I 200



NIF - NOT IDENTIFIABLE
n = 500 I 200

TOTAL : (500+200)X7 = 4900 DATASET

Image Data → from 170.000 → 7000 → 4900

Referensi Dataset :

- Matek, C., Krappe, S., Münzenmayer, C., Haferlach, T., & Marr, C. (2021). An Expert-Annotated Dataset of Bone Marrow Cytology in Hematologic Malignancies [Data set]. The Cancer Imaging Archive. https://doi.org/10.7937/TCIA.AXH3-T579
- Matek, C., Krappe, S., Münzenmayer, C., Haferlach, T., and Marr, C. (2021). Highly accurate differentiation of bone marrow cell morphologies using deep neural networks on a large image dataset. https://doi.org/10.1182/blood.2020010568
- D. Roy, "bone-marrow-cell-classification," Kaggle.com, 2021, doi: https://doi.org/10.7937/TCIA.AXH3-T579.

ABOUT DATASET

Dataset yang digunakan merupakan data gambar dari sel darah putih dibawah mikroskop yang sudah dipisahkan ke dalam 7 kelompok berdasarkan kelompok jenis sel (6) dan satu mewakili kelompok sel tidak terdeteksi.

Cancer Imaging Archive - Munich Leukemia Laboratory (MLL)

We load the data from our own Google Drive
500 Training 200 Test

DATA LOADING

EDA (EXPLORATORY DATA ANALYSIS)

Pre-processing

COUNT, FILE FORMAT

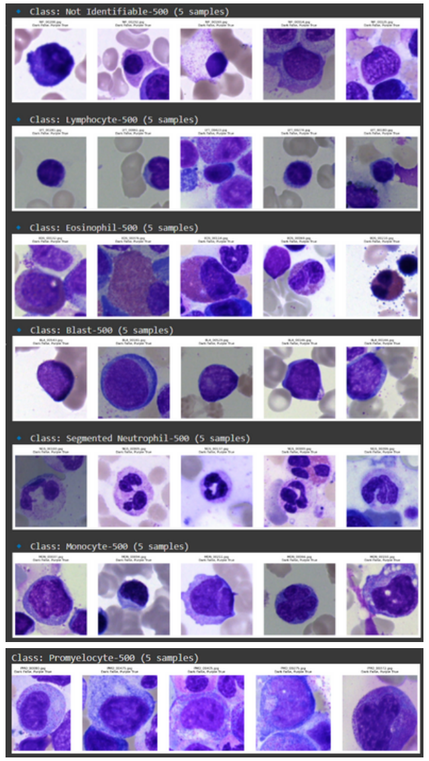
- image_count
- corrupt_files = other than png, jpg, jpeg
- get_color_type = Grayscale/RGB

```
📊 Dataset Quality Report:
📦 Total valid images analyzed   : 3500
❌ Corrupt or unreadable files   : 0
🌑 Too dark images               : 0
🟣 Non-purple-stained images     : 0

🌈 Image Type Distribution:
  - RGB: 3500
```

SAMPLE PREVIEW

- 5 Samples per Class



DARK IMAGE, PURPLE STAINING

- detect_dark_image : 100%
- detect_purple_staining
- source stain : May-Grünwald-Giemsa/Pappenheim stain

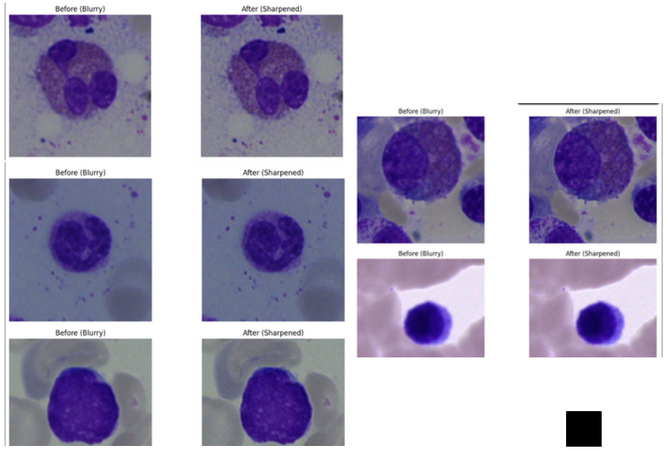
```
📊 Dataset Quality Report:
📦 Total valid images analyzed   : 3500
❌ Corrupt or unreadable files   : 0
🌑 Too dark images               : 0
🟣 Non-purple-stained images     : 0

🌈 Image Type Distribution:
  - RGB: 3500
```

```
📊 EDA Summary:
🔢 Total images: 3500
🟡 Blurry images: 1050 (30.0%)
📏 Avg. image size (px): 62500
⬜ Avg. aspect ratio (W/H): 1.00
📈 Avg. Laplacian variance: 38.12
```

HANDLING BLURRY IMAGES

- Detect with Laplacian variance
- Implementing Unsharp Masking
- Visualization



TRAINING MODEL

Rex-Net 150

Model Convolutional Neural Network (CNN) yang dapat digunakan dalam *image classifiction* secara efisien dan akurat, yang mengadopsi struktur model ResNet dengan memperkenalkan *channel ratio regularization* untuk mengontrol *channels* di setiap layer.

model Rex-Net 150 diambil menggunakan timm.create.model yang tersedia pada library pytorch dengan bobot pre-trained yang menyesuaikan jumlah kelas output (7 kelas).

menggunakan Optimizer berupa Adam yang berbasis *gradient descent* yang memberikan learning rate secara otomatis (default). Loss Function yang distribusi probabiitas prediksi sehingga memaksimalkan proses klasifikasi

menggunakan parameter berupa DataLoader untuk training dan validasi, jumlah epoch sebesar 10 dan learning rate secara default serta patience untuk menggambarkan jumlah epoch toleransi sebelum early stopping

patience membantu menjaga efisiensi training dan mencegah terjadinya overfitting dengan menghentikan training lebih awal ketika model tidak menunjukkan peningkatan kinerja pada beberapa epoch

```
!pip install torchmetrics
import os
import torch
import timm
from tqdm import tqdm
import torchmetrics
from torchmetrics.classification import MulticlassF1Score

class SimpleTrainer:
    def __init__(self, model_name, classes, tr_dl, val_dl, device, epochs=10, lr=3e-4, patience=3, save_path="best_model.pth"):
        self.device = device
        self.model = timm.create_model(model_name, pretrained=True, num_classes=len(classes)).to(device)
        self.tr_dl, self.val_dl = tr_dl, val_dl
        self.loss_fn = torch.nn.CrossEntropyLoss()
        self.optimizer = torch.optim.Adam(self.model.parameters(), lr=lr)
        self.f1 = MulticlassF1Score(num_classes=len(classes)).to(device)
        self.epochs = epochs
        self.patience = patience
        self.save_path = save_path

        self.best_f1, self.no_improve = 0, 0
        self.history = {k: [] for k in ["train_loss", "val_loss", "train_acc", "val_acc", "train_f1", "val_f1"]}

    def run_epoch(self, train=True):
        self.model.train() if train else self.model.eval()
        dl = self.tr_dl if train else self.val_dl
        total_loss, total_correct, total_samples = 0, 0, 0
        self.f1.reset()

        for x, y in tqdm(dl, desc="Train" if train else "Val"):
            x, y = x.to(self.device), y.to(self.device)

            with torch.set_grad_enabled(train):
                pred = self.model(x)
                loss = self.loss_fn(pred, y)
                if train:
                    self.optimizer.zero_grad()
                    loss.backward()
                    self.optimizer.step()

            total_loss += loss.item() * y.size(0)
            total_correct += (pred.argmax(1) == y).sum().item()
            total_samples += y.size(0)
            self.f1.update(pred, y)

        avg_loss = total_loss / total_samples
        acc = total_correct / total_samples
        f1_score = self.f1.compute().item()

        return avg_loss, acc, f1_score
```

```
    def train(self):
        for epoch in range(self.epochs):
            print(f"\nEpoch {epoch+1}/{self.epochs}")
            tr_loss, tr_acc, tr_f1 = self.run_epoch(train=True)
            val_loss, val_acc, val_f1 = self.run_epoch(train=False)

            print(f"Train -> Loss: {tr_loss:.4f}, Acc: {tr_acc:.4f}, F1: {tr_f1:.4f}")
            print(f"Val   -> Loss: {val_loss:.4f}, Acc: {val_acc:.4f}, F1: {val_f1:.4f}")

            for k, v in zip(self.history, [tr_loss, val_loss, tr_acc, val_acc, tr_f1, val_f1]):
                self.history[k].append(v)

            if val_f1 > self.best_f1:
                self.best_f1 = val_f1
                torch.save(self.model.state_dict(), self.save_path)
                print(f"✅ Saved best model (F1: {val_f1:.4f})")
                self.no_improve = 0
            else:
                self.no_improve += 1
                print(f"⚠️ No improvement ({self.no_improve}/{self.patience})")
                if self.no_improve >= self.patience:
                    print("🛑 Early stopping")
                    break

# Define the device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

trainer = SimpleTrainer("rexnet_150", classes, tr_dl, val_dl, device)
trainer.train()
```

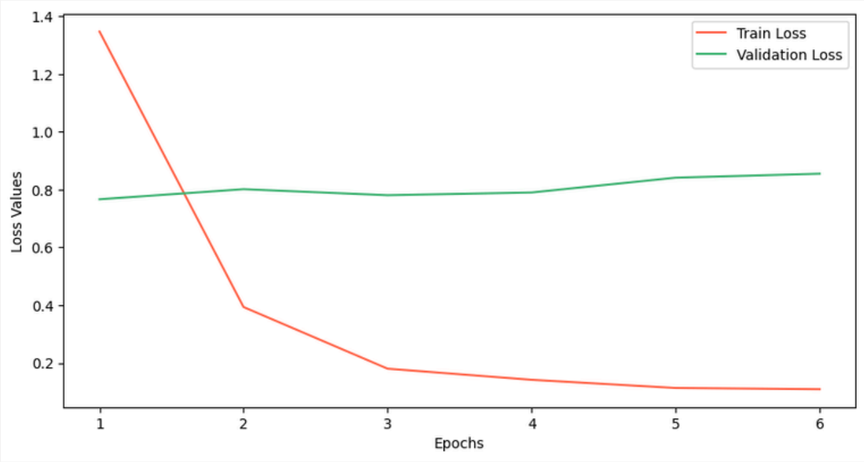
RESULTS

dataset Snapshot and EDA

Loss adalah ukuran seberapa jauh prediksi model dari hasil yang sebenarnya.

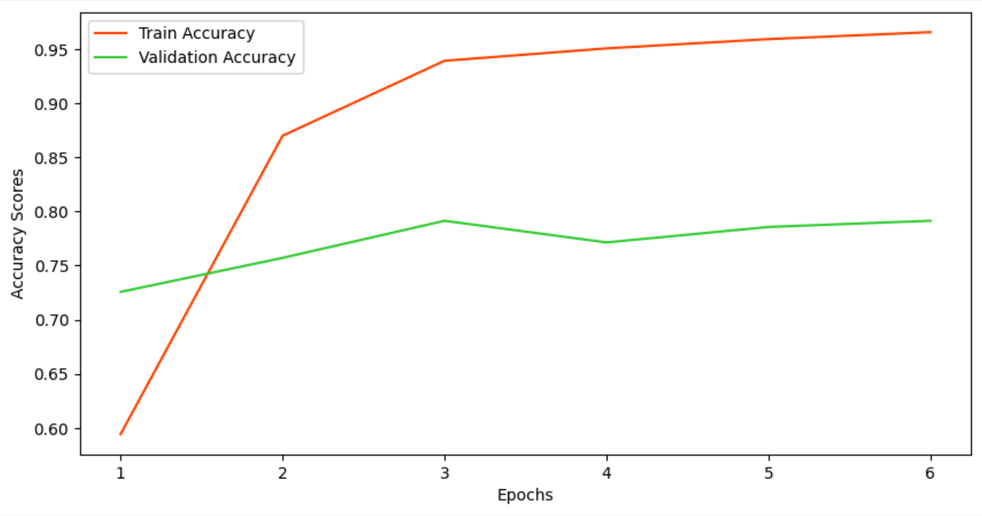
Accuracy adalah persentase prediksi model yang benar dari seluruh data.

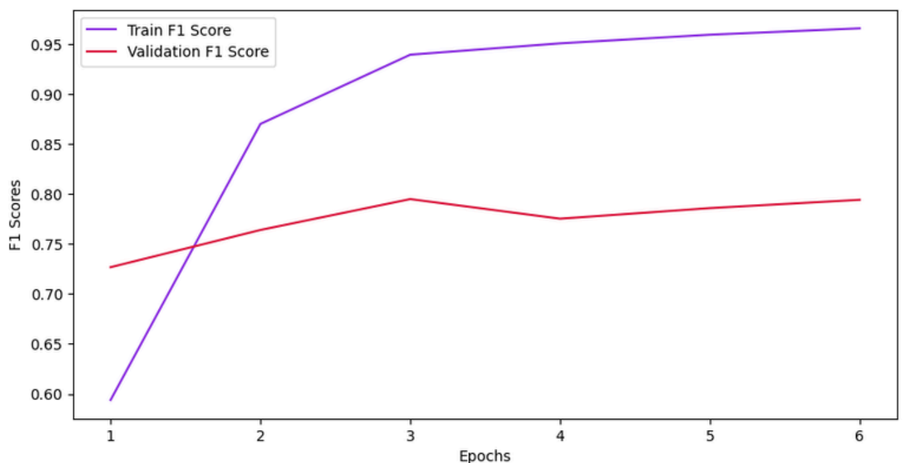
F1 Score adalah metrik gabungan dari Precision dan Reca



LOSS VALUE

ACCURACY SCORE

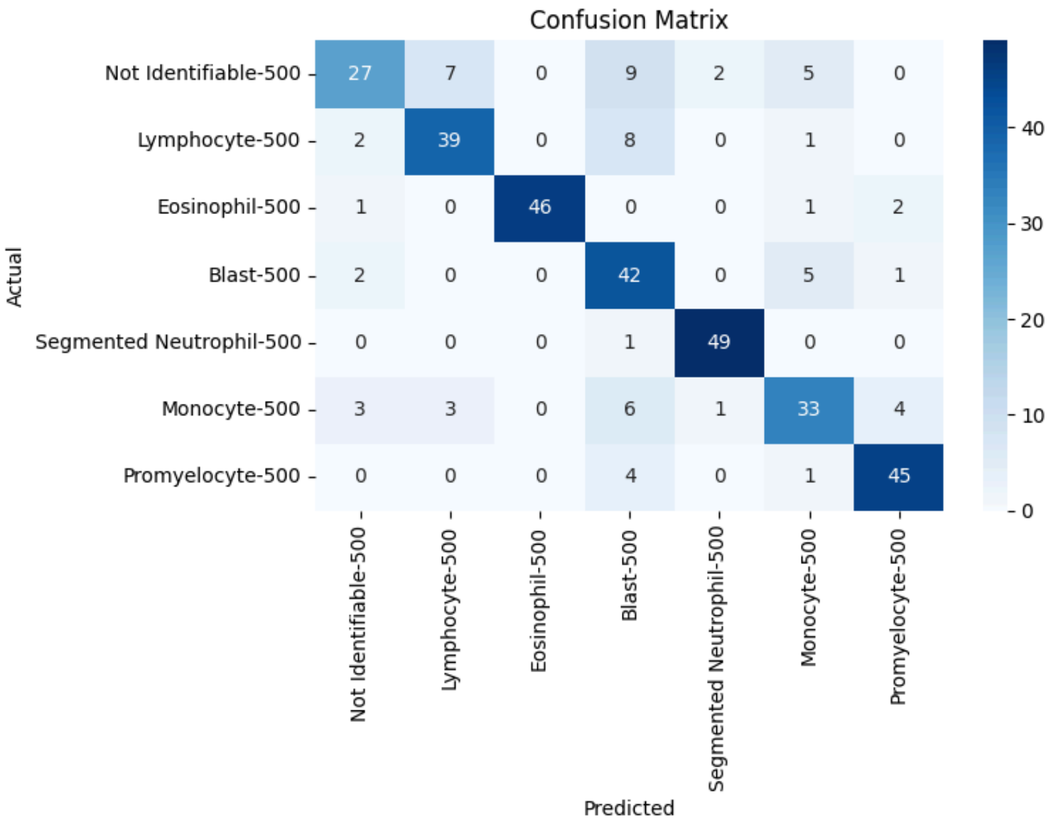




F1 SCORE

PROGRAM STUDI TEKNIK BIOMEDIK

RESULTS



Akurasi : 80,29%

F1-Score : 80,29%

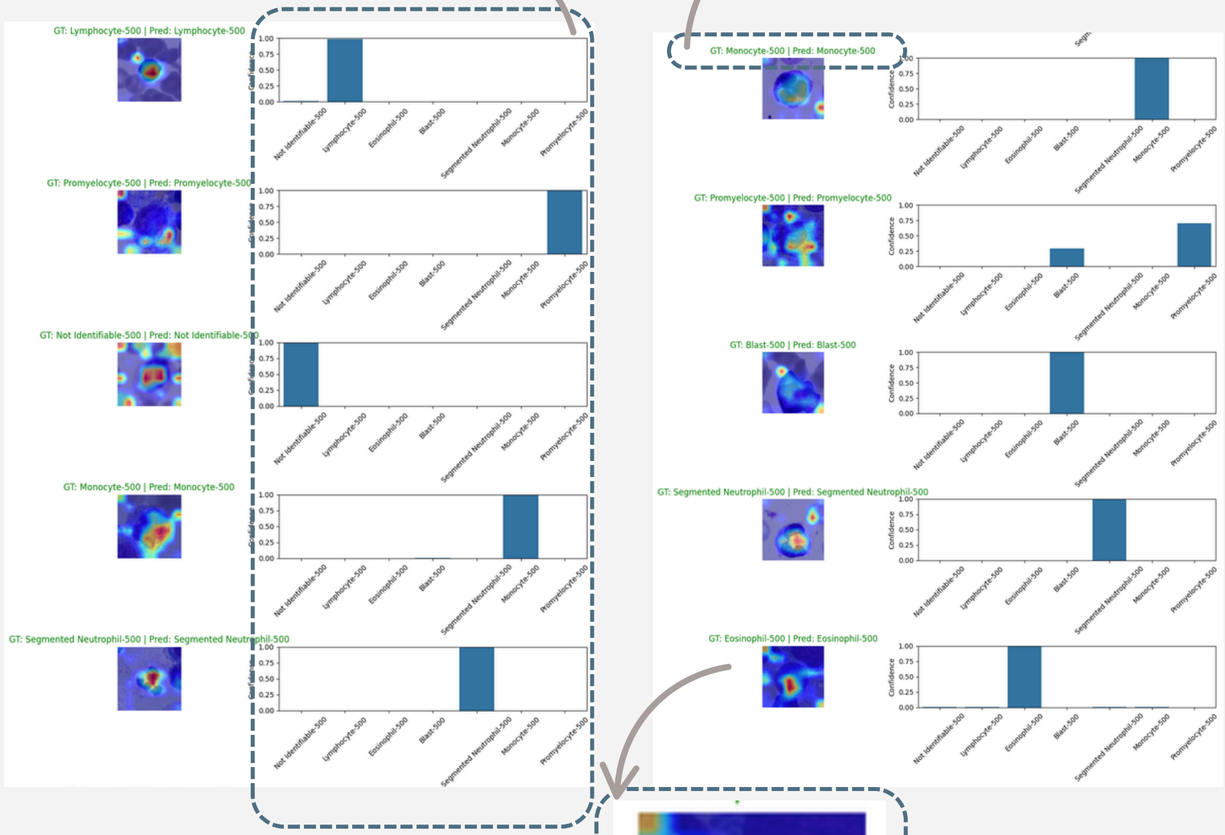
Model RexNet-150 melakukan **klasifikasi** dengan **cukup baik**, Eosinophil, Segmented Neutrophil dan Promyelocyte.

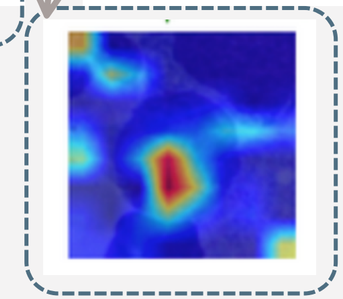
Model masih mengalami **kesalahan** yang cukup signifikan pada kelas **Not Identifiable** dan **Monocyte** akibat adanya **kemiripan morfologis** dan **kualitas gambar**

Penampilan **Bar Plot** yang menunjukkan **klasifikasi** sel sumsum tulang belakang, **tinggi bar** sebanding dengan keyakinan (*confidence* score)

Klasifikasi dilakukan pada *confidence score* **paling tinggi** (ideal = 100%) dari serangkaian kelas lainnya

Kesesuaian klasifikasi model antara nilai ***actual* and *predicted*** dari, adanya kebenaran klasifikasi tapi mungkin memiliki *confidence score* yang moderat

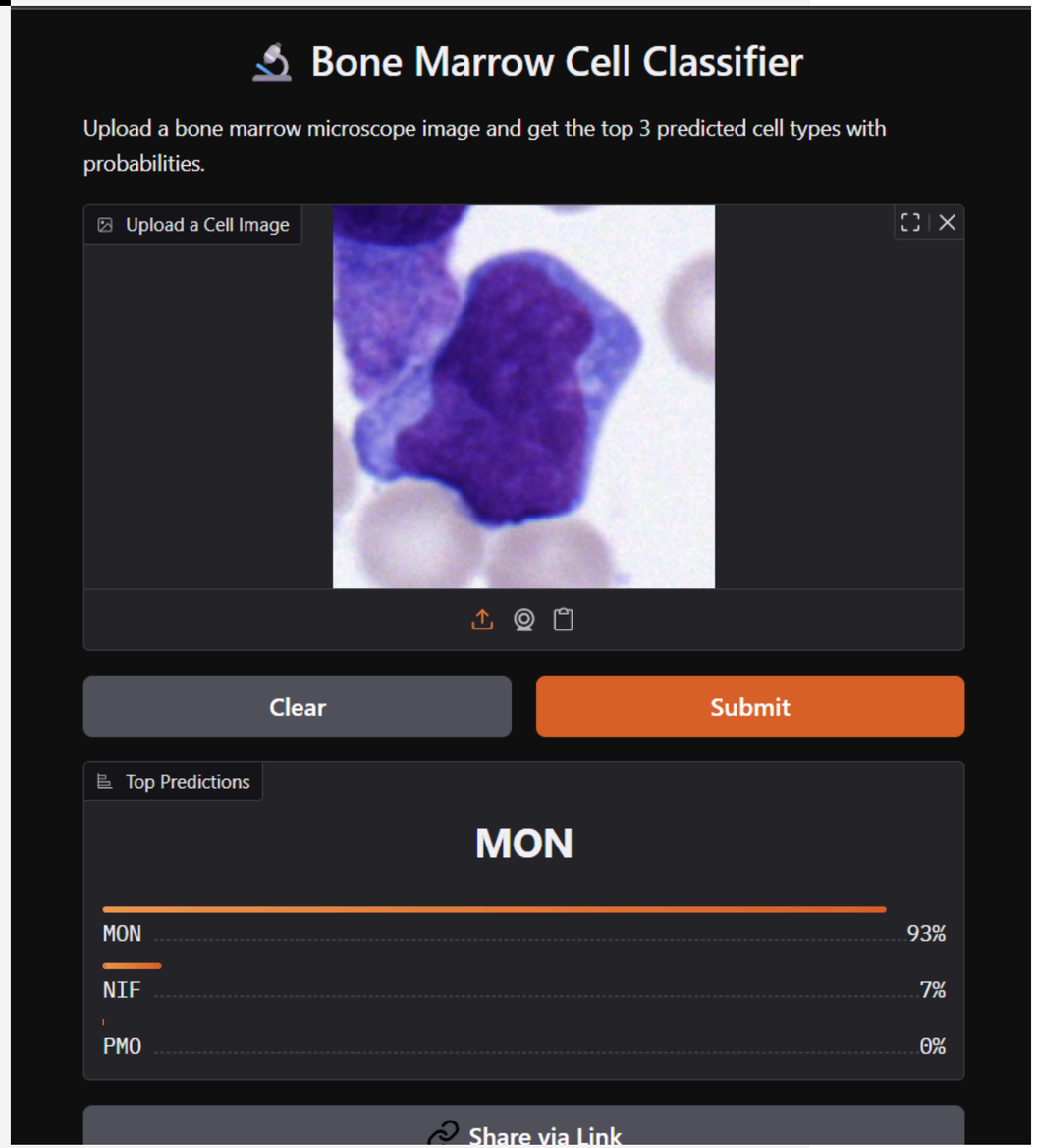




GRAD-CAM

Teknik **Visualisasi** berupa **heatmap** untuk memahami *features* (area penting) yang paling **berkontribusi/berkaitan** terhadap model *training* dalam melalukan **klasifikasi** gambar.

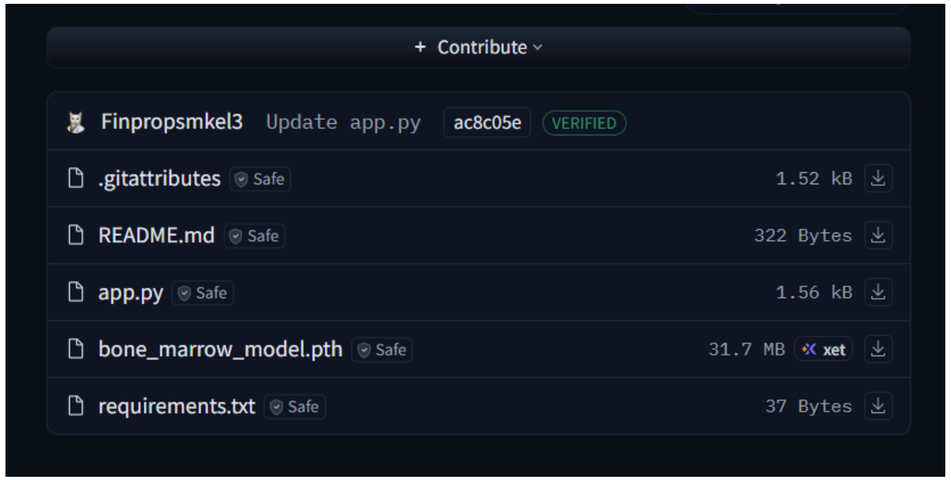
PROGRAM STUDI TEKNIK BIOMEDIK



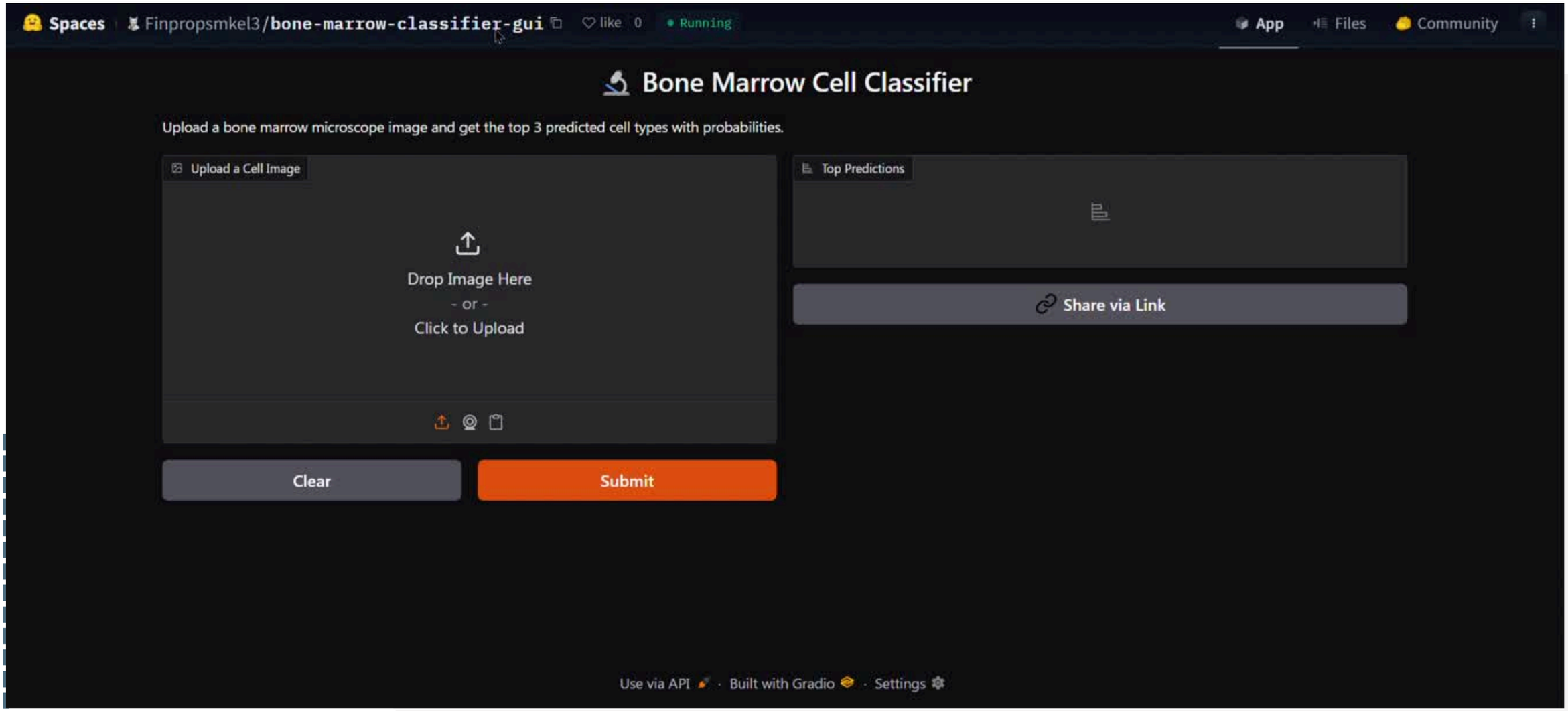
GUI

APA YANG DILAKUKAN

- Mengambil model .pth dari Colab untuk mengklasifikasikan sel sumsum tulang.
- Memproses gambar yang diunggah.
- Memprediksi 3 jenis sel yang paling mungkin.
- Menampilkan hasilnya dalam *interface* menggunakan Gradio.



GUI DEMO


Spaces
Finpropsmkel3/bone-marrow-classifier-gui
like 0
Running
App
Files
Community
Bone Marrow Cell Classifier
Upload a bone marrow microscope image and get the top 3 predicted cell types with probabilities.
Upload a Cell Image
Drop Image Here
- or -
Click to Upload
Top Predictions
Share via Link
Clear
Submit
Use via API · Built with Gradio · Settings

LIMITATIONS, TAKE-HOME MESSAGE, FUTURE WORK

KETERBATASAN

- Model hanya dilatih pada 7 kelas dari total 21 kelas.
- Akurasi sedikit menurun untuk tipe sel yang langka atau mirip, seperti faggot cells atau abnormal eosinophils.
- Hanya dapat melakukan single label classification

ARAH PENGEMBANGAN

- Perluasan ke 21 kelas sel
- Melakuakan multi label classification
- Melakukan prediksi dari penyakit hematologi

TAKE HOME MESSAGE

Deep learning alat untuk mengotomatisasi klasifikasi morfologi sel sumsum tulang, menawarkan akurasi, konsistensi, dan efisiensi — serta berpotensi besar sebagai diagnosis yang luas.

KELOMPOK 4

THANK YOU