# Tissue Flow AI Trenowanie modelu – Kompletna instrukcja (jednym ciągiem)

Ta wersja działa w Colabie – wystarczy uruchomić wszystko od góry do dołu. Plik ZIP należy wgrać ręcznie przez ikonę po lewej stronie (Pliki → Prześlij).

## 🔧 Kompletna wersja kodu:

# Krok 1: Wgrywanie i wypakowanie ZIP-a z lokalnego komputera

import zipfile

import os

import matplotlib.pyplot as plt

import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.models import Model

from tensorflow.keras.layers import GlobalAveragePooling2D, Dense

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

uploaded\_path = '/content/tissue\_flow\_ai\_demo\_final.zip' # Wgraj ten plik ręcznie do Colab

with zipfile.ZipFile(uploaded\_path, 'r') as zip\_ref:

zip\_ref.extractall('/content/tissue\_flow\_ai\_demo')

# Krok 2: Ustawienie ścieżki do folderu z obrazami

base\_dir = '/content/tissue\_flow\_ai\_demo/images'

# Krok 3: Przygotowanie danych treningowych i walidacyjnych

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

train\_data = datagen.flow\_from\_directory(

base\_dir,

target\_size=(224, 224),

batch\_size=4,

class\_mode='binary',

subset='training',

shuffle=True

)

val\_data = datagen.flow\_from\_directory(

base\_dir,

target\_size=(224, 224),

batch\_size=4,

class\_mode='binary',

subset='validation',

shuffle=False

)

# Krok 4: Budowa modelu z MobileNetV2

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

base\_model.trainable = False

x = base\_model.output

x = GlobalAveragePooling2D()(x)

predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Krok 5 – Trening modelu z przechwyceniem historii

history = model.fit(train\_data, validation\_data=val\_data, epochs=5)

# Krok 6 – Wizualizacja przebiegu treningu – wykresy accuracy i loss

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Val Accuracy')

plt.title('Dokładność klasyfikacji')

plt.xlabel('Epoka')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

plt.show()

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Val Loss')

plt.title('Strata (loss)')

plt.xlabel('Epoka')

plt.ylabel('Loss')

plt.legend()

plt.grid(True)

plt.show()

# Krok 7 – Predykcja i wyświetlenie kilku obrazów z etykietami

val\_data.reset()

sample\_images, labels = next(val\_data)

predictions = model.predict(sample\_images)

class\_names = list(val\_data.class\_indices.keys())

num\_samples = len(sample\_images)

for i in range(num\_samples):

plt.imshow(sample\_images[i])

plt.title(f"Prawda: {class\_names[int(labels[i])]} | Pred: {class\_names[int(predictions[i][0] > 0.5)]}")

plt.axis('off')

plt.show()

# Krok 8 – Macierz pomyłek

val\_data.reset()

y\_true = val\_data.classes

y\_pred = (model.predict(val\_data) > 0.5).astype("int32")

cm = confusion\_matrix(y\_true, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=class\_names)

disp.plot(cmap=plt.cm.Blues)

plt.title("Macierz pomyłek (confusion matrix)")

plt.show()

🎯 Gotowe! Wystarczy uruchomić wszystko po kolei w Colabie. Plik ZIP przesyłasz klikając po lewej stronie w zakładkę „Pliki” → „Prześlij”.