ДОДАТОК А

Програмний код тренування моделей

```
img_shape=(img_size[0], img_size[1], 3)
n_classes = 6
model_name='EfficientNetB0'
base_model= EfficientNetB0(include_top=False, weights="imagenet",input_shape=img_shape, pooling='max')

base_model.trainable=False
x=base_model.output
x=BatchNormalization(axis=-1, momentum=0.999, epsilon=0.001 )(x)
x = Dense(1024, kernel_regularizer = regularizers.l2(0.01),activity_regularizer=regularizers.l1(0.005),
                bias_regularizer=regularizers.l1(0.005) ,activation='relu')(x)
x=Dropout(rate=.2, seed=123)(x)
x = Dense(512, kernel_regularizer = regularizers.l2(0.01),activity_regularizer=regularizers.l1(0.005),
                bias_regularizer=regularizers.l1(0.005) ,activation='relu')(x)
x=Dropout(rate=.3, seed=123)(x)
x = Dense(256, kernel_regularizer = regularizers.l2(0.01),activity_regularizer=regularizers.l1(0.005),
                bias_regularizer=regularizers.l1(0.005) ,activation='relu')(x)
x=Dropout(rate=.4, seed=123)(x)
output=Dense(n_classes, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
lr=.0001

model=Sequential()
model.add(base_model)
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(512, activation='elu'))
model.add(Dense(256, activation='elu'))
model.add(Dense(128, activation = 'elu'))
model.add(Dense(6, activation='softmax'))
```

Рисунок А1 – Програмний код створення моделі на основі EfficientNetB0

```
Epoch 1/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 537s 4s/step - acc: 0.4199 - loss: 1.7029 - val_acc: 0.8048 - val_loss: 0.5588
Epoch 2/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - acc: 0.6991 - loss: 0.8207📁 History saved at epoch 2
120/120 ━━━━━━━━━━━━━━━━━━━━ 523s 4s/step - acc: 0.6993 - loss: 0.8202 - val_acc: 0.8539 - val_loss: 0.4196
Epoch 3/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 477s 4s/step - acc: 0.7741 - loss: 0.6099 - val_acc: 0.8643 - val_loss: 0.3746
Epoch 4/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - acc: 0.8001 - loss: 0.5664WARNING:absl:You are saving your model as

✅ Saved model checkpoint: /content/drive/MyDrive/diploma/saves_transfer/model_epoch_04.weights.h5
📁 History saved at epoch 4
120/120 ━━━━━━━━━━━━━━━━━━━━ 527s 4s/step - acc: 0.8003 - loss: 0.5660 - val_acc: 0.8862 - val_loss: 0.3157
Epoch 5/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 527s 4s/step - acc: 0.8120 - loss: 0.5095 - val_acc: 0.8956 - val_loss: 0.2895
Epoch 6/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - acc: 0.8316 - loss: 0.4681📁 History saved at epoch 6
120/120 ━━━━━━━━━━━━━━━━━━━━ 480s 4s/step - acc: 0.8316 - loss: 0.4680 - val_acc: 0.8967 - val_loss: 0.2744
Epoch 7/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 537s 4s/step - acc: 0.8519 - loss: 0.4188 - val_acc: 0.9040 - val_loss: 0.2538
Epoch 8/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - acc: 0.8537 - loss: 0.3948WARNING:absl:You are saving your model as

✅ Saved model checkpoint: /content/drive/MyDrive/diploma/saves_transfer/model_epoch_08.weights.h5
📁 History saved at epoch 8
120/120 ━━━━━━━━━━━━━━━━━━━━ 550s 4s/step - acc: 0.8537 - loss: 0.3946 - val_acc: 0.9102 - val_loss: 0.2475
Epoch 9/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 542s 5s/step - acc: 0.8608 - loss: 0.3764 - val_acc: 0.9144 - val_loss: 0.2408
Epoch 10/30
120/120 ━━━━━━━━━━━━━━━━━━━━ 0s 3s/step - acc: 0.8657 - loss: 0.3638📁 History saved at epoch 10
120/120 ━━━━━━━━━━━━━━━━━━━━ 552s 5s/step - acc: 0.8658 - loss: 0.3636 - val_acc: 0.9144 - val_loss: 0.2364
```

Рисунок А2 – Початок тренування моделі трансферного навчання на EfficientNetB0

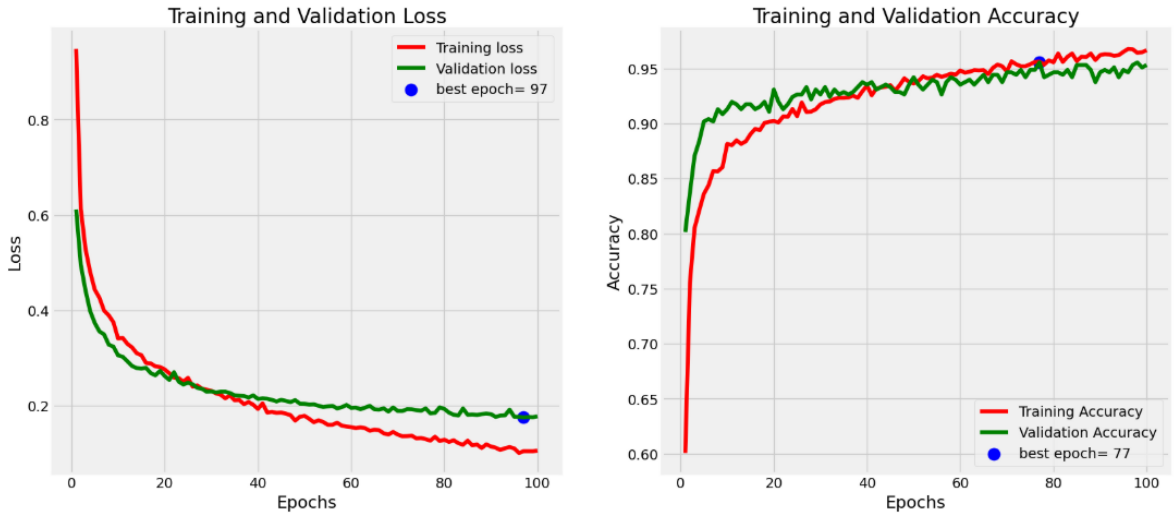

Рисунок А3 – Графік функції втрат та точності моделі трансферного навчання

```
pred_probs = model.predict(X_val)
y_pred = np.argmax(pred_probs, axis=1)

n_classes = len(class_names)
print("Accuracy:", accuracy_score(y_true, y_pred))
print(classification_report(y_true, y_pred, target_names=class_names, digits=4))

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names, yticklabels=class_names, cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()

sensitivities = np.zeros(n_classes)
specificities = np.zeros(n_classes)

total = cm.sum()

for i in range(n_classes):
    TP = cm[i, i]
    FN = cm[i, :].sum() - TP
    FP = cm[:, i].sum() - TP
    TN = total - TP - FP - FN

    sensitivities[i] = TP / (TP + FN) if (TP + FN) > 0 else float('nan')
    specificities[i] = TN / (TN + FP) if (TN + FP) > 0 else float('nan')

print("\nPer-class Sensitivity (Recall) and Specificity:")
for i, name in enumerate(class_names):
    print(f"{name:25s}  Sensitivity: {sensitivities[i]:.4f}   Specificity: {specificities[i]:.4f}")

macro_sens = np.nanmean(sensitivities)
macro_spec = np.nanmean(specificities)
balanced_acc = balanced_accuracy_score(y_true, y_pred)

print(f"\nMacro Sensitivity (Recall): {macro_sens:.4f}")
print(f"Macro Specificity:         {macro_spec:.4f}")
print(f"Balanced Accuracy:         {balanced_acc:.4f}")
```

Рисунок А4 – Код функції тестування точності моделей

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	74 с.
	Інші документи	
2.	Презентаційні матеріали	18 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Гур'єв М.І.			Розробка та дослідження методів машинного навчання в задачах аналізу біомедичних зображень для визначення патологій	Шифр групи	Код напр./спец.
Перевірив		Філатов В.О.				ДСм-24-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					