המכללה האקדמית להנדסה בראודה בכרמיאל

המחלקה להנדסת חשמל ואלקטרוניקה

מערכות לומדות ולמידה עמוקה

סמסטר ב – תשפ"ה 2025

מטלת סוף

משימת הפרוייקט: גילוי דלקת ריאות בצילומי רנטגן

בשארה חביב

פרנסיס עבוד

מריה נחלה

שם המרצה: פרופ' אמיר אדלר

תאריך: 23/08/2025

תוכן העניינים

1) תוכן העניינים 2
2) מבוא - הגדרת העבודה 4
3) משימה 1 – תכנון רשתות CNN 5
הדמיית תמונות מהדאטהסט 5
רשת 1 – ללא Transfer Learning (רשת מותאמת אישית) 6
רשת עמוקה 2 : (CNN with Transfer Learning) ..... Error! Bookmark not defined.
רשת עמוקה (CNN with Transfer Learning and Fine Tuning) : 2B 11
4) משימה 2 – אימון והערכת ביצועים של מודלים 4
ייבוא הנתונים והצגה ראשונית 13
אימון מודלים והשוואת שיטות Transfer Learning – לעומת Fine Tuning ..... Error! Bookmark not defined.
רשת 1 – ללא Transfer Learning 14
רשת 2– Transfer Learning (Frozen) 15
רשת 3 – Transfer Learning (Fine-Tuned) 16
Precision-Recall ותלות בסף החלטה ..... Error! Bookmark not defined.
בחירת מודל מנצח לפי F1 ..... Error! Bookmark not defined.
5) משימה 3 – בדיקת השפעת אלגוריתם האופטימיזציה והפעלת Early Stopping 21
השפעת Optimizer, Learning Rate ו- Epochs 22
א. אלגוריתם SGD 22
➢ SGD - Graph for learning rate 0.01 22
➢ SGD - Graph for learning rate 0.001 25
➢ SGD - Graph for learning rate 0.0001 28
ב. אלגוריתם SGD עם Momentum 31
➢ SGD with Momentum - Graph for learning rate 0.01 31
➢ SGD with Momentum - Graph for learning rate 0.001 34
➢ SGD with Momentum - Graph for learning rate 0.0001 37
ג. אלגוריתם Adam 40
➢ Adam - Graph for learning rate 0.01 40
➢ Adam - Graph for learning rate 0.001 43
➢ Adam - Graph for learning rate 0.0001 46
ד. אלגוריתם RMSprop 49
➢ RMSprop - Graph for learning rate 0.01 49

➢ RMSprop - Graph for learning rate 0.001 52
➢ RMSprop - Graph for learning rate 0.0001 55
השוואת הביצועים ובחירת Optimizer מנצח ..... Error! Bookmark not defined.
ה. הפעלת Early Stopping 67
6) משימה 4 – סיווג מרובה קטגוריות באמצעות CNN 70
אימון הרשת עם אופטימייזרים שונים 70
א. אלגוריתם SGD 71
➢ SGD - Graph for learning rate 0.01 71
➢ SGD - Graph for learning rate 0.001 73
➢ SGD - Graph for learning rate 0.0001 74
ב. אלגוריתם SGD עם Momentum 76
➢ SGD with Momentum - Graph for learning rate 0.01 76
➢ SGD with Momentum - Graph for learning rate 0.001 77
➢ SGD with Momentum - Graph for learning rate 0.0001 79
ג. אלגוריתם Adam 80
➢ Adam - Graph for learning rate 0.01 80
➢ Adam - Graph for learning rate 0.001 82
➢ Adam - Graph for learning rate 0.0001 83
ד. אלגוריתם RMSprop 85
➢ RMSprop - Graph for learning rate 0.01 85
➢ RMSprop - Graph for learning rate 0.001 86
➢ RMSprop - Graph for learning rate 0.0001 88
סיכום השוואתי בין כל הקונפיגורציות 89
Confusion Matrix לסיווג תלת־קטגורי 91
סיכום של משימה 4 91
7) סיכום כללי – דוח פרויקט: גילוי דלקת ריאות בצילומי רנטגן 92
מהלכי העבודה המשותפים לכל המשימות 92
שיאי הביצועים (Highlights) 92
מסקנות מרכזיות 92

משימה 2 – אימון והערכת ביצועים של מודלים

מבוא - הגדרת העבודה

בפרויקט ימומשו פתרונות מבוססי CNN לגילוי דלקת ריאות מ- 5863 צילומים אמיתיים הכוללים אבחון של רופאים. התמונות נמצאות בקישור הבא.
https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
להלן מספר דוגמאות משתי הקטגוריות:

בנוסף, קיימת הפרדה לשני סוגי דלקת ריאות: חיידקית (Bacterial) ונגיפית (Viral), להלן דוגמאות:

מטרת הפרויקט היא תכנון רשתות למידה עמוקה לגילוי דלקת ריאות, כאשר הביצועים ימדדו על סט תמונות בדיקה, אשר יופרד באופן מוחלט מסט האימון. ביצועים טובים יחשבו לסיכוי שגיאה נמוך מ-7%- (כלומר Accuracy גבוה מ - 93%). יש להשתמש בסט תמונות בדיקה הכולל לפחות 200 תמונות עם דלקת ריאות (100 מכל סוג) ו 200 - תמונות ללא דלקת ריאות. יש ליצור סט אימון ובדיקה חדשים מתוך כלל התמונות הקיימות באתר, ולא לפי החלוקה הקיימת באתר. סטים חדשים אלה ישמשו את כל סעיפי הפרויקט.

משימה 1 – תכנון רשתות CNN

עליכם לתכנן שתי רשתות עמוקות שונות לפתרון הבעייה: כניסת כל רשת עמוקה היא תמונה והמוצא הינו ההסתברות שהתמונה מייצגת מקרה חיובי של דלקת ריאות (בסעיף זה אין זה משנה מה סוג הדלקת -חיידקית או נגיפית). ההסתברות היא בין 0 ל-1 כאשר 1 מייצג מקרה וודאי של דלקת ריאות.

רשת ראשונה: רשת CNN ללא שימוש ב- TRANSFER LEARNING.

רשת שניה: רשת CNN מבוססת TRANSFER LEARNING.

מותר להיעזר בהצעות לפתרונות המופיעים בקישור הבא (מדובר על הצעות של אנשים פרטיים ולא פתרון "רשמי"):

https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia/code

יש לפרט בדוח הפרוייקט את מבנה שתי הרשתות שתכננתם, כולל כל השכבות. לגבי הרשת המבוססת TRANSFER LEARNING יש לפרט את שם ומבנה רשת הבסיס (שכבר אומנה בעבר), איזה שכבות נלקחו ממנה, ואיזה שכבות הוספתם להשלמת הפתרון הנדרש.

הדמיית תמונות מהדאטהסט

לפני בניית הרשתות, ביצענו הדמיה של 8 תמונות לדוגמה מהדאטהסט. ניתן להבחין בהבדל בין צילומים תקינים לצילומים עם דלקת ריאות.

רשת 1 – ללא Transfer Learning (רשת מותאמת אישית)

רשת זו נבנתה באופן ידני וללא שימוש במודלים מוכנים) כגון ResNet או VGG), כלומר כל שכבותיה הוגדרו מאפס (from scratch). הרשת כוללת ארבעה בלוקים זהים של שכבות Convolution־ MaxPooling, שכל אחד מהם מיועד לחלץ תכונות רלוונטיות מהתמונה ולהפחית את ממדיה באופן הדרגתי. מטרת המבנה הזה היא לאפשר לרשת ללמוד ייצוגים היררכיים – מקווים פשוטים ועד אזורים מורכבים יותר – לפני המעבר לשלב הסיווג הסופי.
הרשת פותחה עם מטרת סיווג בינארי (דלקת ריאות לעומת מצב תקין), תוך שימוש בשכבת פלט אחת עם פונקציית אקטיבציה מסוג Sigmoid. פונקציית האיבוד שנבחרה היא Binary Crossentropy ואופטימיזר מסוג Adam שימש לעדכון המשקלים במהלך האימון.

הרכב השכבות:

1) **Rescaling**:
משנה את ערך של הפיקסלים בתמונה מ-[0-255] ל- [0-1]

2) **Conv2D:**
שכבה שיכולה לשמש כשכבה עליונה ברשת, כפי שהוסבר קודם, השכבה לוקחת את התמונה הדו ממדית ומבצעת פעולות כמו סינון, זיהוי קצוות וכו'.
בחירת גדלים:
- מספר מסננים 32
- גודל ה- KERNEL שווה ל- 3X3
- פונקצית האקטיבציה של השכבה "Relu"

3) **Maxpooling2D**:
תפקיד השכבה הוא להקטין את עלות החישוב (מקטינה את ממדי המטריצה הדו ממדית שנכנסת לשכבה).
בחירת גדלים:
- גודל חלון: 2×2 (כלומר Strides = 2)

4) **Flatten:**
משתמשים בשכבה זו ליצירת פלט שטוח. המטרה משכבה זו הוא להפוך מטריצה דו ממדית של תכונות לווקטור שניתן להכניס אותו לשכבת CNN Classifier.

5) **Dense:**
שכבה שמייצגת כפל וקטורים במטריצה. הערכים במטריצה הם הפרמטרים הניתנים לאימון המתעדכנים במהלך ההפצה.
- שכבה ראשונה עם 128 נוירונים (ReLU)
- שכבת פלט עם נוירון 1 (Sigmoid)

Code:

```python
def create_cnn_without_transfer_learning():
    """
    SOLUTION FOR QUESTION 1A: Custom CNN Architecture
    Report Section: Network 1 - CNN without Transfer Learning

    Creates a custom CNN model from scratch for binary pneumonia classification.

    ARCHITECTURE DETAILS:
    - Input Layer: 160x160x3 RGB chest X-ray images
    - Convolutional Blocks: 4 identical blocks with:
      * Conv2D: 32 filters, 3x3 kernel, ReLU activation
      * MaxPooling2D: 2x2 pool size, stride=2
    - Feature Extraction: Flatten layer converts 3D to 1D
    - Classification Head:
      * Dense: 128 neurons, ReLU activation
      * Output: 1 neuron, Sigmoid activation (binary classification)

    TRAINING CONFIGURATION:
    - Optimizer: Adam (default learning rate)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy

    REPORT USAGE: Include architecture diagram and parameter count
    """
```

```python
model = Sequential([
    # Explicit input with fixed batch size for clearer diagrams
    tf.keras.Input(shape=(*IMG_SIZE, 3)),
    # First Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Second Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Third Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Fourth Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Feature Extraction
    Flatten(),

    # Classification Head
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification output
])

# Compile model with binary classification settings
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

return model
```

Summary:

CNN without Transfer Learning

Layer (Type)	Output Shape	Parameters
conv2d_4 (Conv2D)	(32, 158, 158, 32)	**896**
max_pooling2d_10 (MaxPooling2D)	(32, 79, 79, 32)	**0**
conv2d_5 (Conv2D)	(32, 77, 77, 32)	**9,248**
max_pooling2d_11 (MaxPooling2D)	(32, 38, 38, 32)	**0**
conv2d_6 (Conv2D)	(32, 36, 36, 32)	**9,248**
max_pooling2d_12 (MaxPooling2D)	(32, 18, 18, 32)	**0**
conv2d_7 (Conv2D)	(32, 16, 16, 32)	**9,248**
max_pooling2d_13 (MaxPooling2D)	(32, 8, 8, 32)	**0**
flatten_3 (Flatten)	(32, 2048)	**0**
dense_6 (Dense)	(32, 128)	**262,272**
dense_7 (Dense)	(32, 1)	**129**

Total Parameters: 291,041 | Trainable: 291,041 | Non-trainable: 0

Total Parameters: 291,041 | Trainable: 291,041

conv2d_4 (Conv2D)
Input shape: (32, 160, 160, 3)
Output shape: (32, 158, 158, 32)
max_pooling2d_10 (MaxPooling2D)
Input shape: (32, 158, 158, 32)
Output shape: (32, 79, 79, 32)
conv2d_5 (Conv2D)
Input shape: (32, 79, 79, 32)
Output shape: (32, 77, 77, 32)
max_pooling2d_11 (MaxPooling2D)
Input shape: (32, 77, 77, 32)
Output shape: (32, 38, 38, 32)
conv2d_6 (Conv2D)
Input shape: (32, 38, 38, 32)
Output shape: (32, 36, 36, 32)
max_pooling2d_12 (MaxPooling2D)
Input shape: (32, 36, 36, 32)
Output shape: (32, 18, 18, 32)
conv2d_7 (Conv2D)
Input shape: (32, 18, 18, 32)
Output shape: (32, 16, 16, 32)
max_pooling2d_13 (MaxPooling2D)
Input shape: (32, 16, 16, 32)
Output shape: (32, 8, 8, 32)
flatten_3 (Flatten)
Input shape: (32, 8, 8, 32)
Output shape: (32, 2048)
dense_6 (Dense)
Input shape: (32, 2048)
Output shape: (32, 128)
dense_7 (Dense)
Input shape: (32, 128)
Output shape: (32, 1)

רשת עמוקה 2 : (CNN with Transfer Learning)

רשת זו עושה שימוש ב־**Transfer Learning** על בסיס **ResNet152V2** – מודל עמוק שאומן מראש על מאגר התמונות. **ImageNet** הרשת נבחרה בזכות הדיוק הגבוה שלה ומשום יכולתה ללמוד ייצוגים חזותיים מורכבים ורב־שכבתיים. השתמשנו **בכל השכבות הקודמות של המודל (כולל כובדיהם)** אך **השמטנו את ה־Top** (שכבת הסיווג המקורית), שכן מטרתנו שונה מזו של **ImageNet** (סיווג ל־1000 קטגוריות) כדי שנוכל להוסיף סיווג מותאם לבעיה שלנו ואנו מבצעים סיווג בינארי.

על בסיס הפלט של **ResNet152V2** הוספנו שלוש שכבות:

1. **Flatten** – הופכת את המטריצה התלת-ממדית של הפלט מהמודל לקלט וקטורי רציף לשכבות הסיווג.
2. **Dense (128 נוירונים,ReLU)**– שכבה צפופה הלומדת תכונות לא ליניאריות ומסייעת בהתאמה לבעיה הספציפית.
3. **Dense (1 נוירון, Sigmoid)**–שכבת הפלט שמבצעת סיווג בינארי (דלקת ריאות/בריא) עם הסתברות כערך בין 0 ל־1.

חשוב לציין שכל שכבות ה־**ResNet** הוגדרו כ־**Frozen** כדי לשמר את הידע הקיים (**Pretrained Features**), ומנענו בכך אימון מחדש שעלול להוביל ל־**Overfitting** על דאטה קטן.

ResNet152V2:

- include_top: whether to include the fully-connected layer at the top of the network.
- weights: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.
- input_shape: optional shape tuple, only to be specified if include_top is False
- pooling: Optional pooling mode for feature extraction when include_top is False.

Code:

```python
def create_cnn_with_transfer_learning_frozen():
    SOLUTION FOR QUESTION 1B (APPROACH 1): Transfer Learning with Frozen Base
    Report Section: Network 2A - CNN with Transfer Learning (Frozen)

    Creates CNN using pre-trained ResNet152V2 with frozen weights for feature extraction.

    ARCHITECTURE DETAILS:
    Base Network:
    - Model: ResNet152V2 pre-trained on ImageNet
    - Input Shape: 160x160x3 RGB images
    - Include Top: False (remove original classification head)
    - Trainable: False (frozen weights - no backpropagation)

    Custom Classification Head:
    - Flatten: Convert ResNet features to 1D vector
    - Dense: 128 neurons, ReLU activation
    - Output: 1 neuron, Sigmoid activation (binary classification)

    TRAINING CONFIGURATION:
    - Optimizer: Adam with low learning rate (0.0001)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy

    ADVANTAGES:
    - Fast training (only classification head is trained)
    - Good for small datasets
    - Leverages ImageNet-learned features

    REPORT USAGE: Compare with custom CNN and fine-tuned approach
```

```python
def create_cnn_with_transfer_learning_frozen():

    # Load pre-trained ResNet152V2 without classification head
    inputs = tf.keras.Input(shape=(*IMG_SIZE, 3), batch_size=BATCH_SIZE)
    base_model = ResNet152V2(
        weights='imagenet',
        include_top=False,
        input_tensor=inputs
    )

    # Freeze all layers in the base model
    base_model.trainable = False

    # Create complete model with custom classification head
    model = Sequential([
        base_model,
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    # Compile with low learning rate for transfer learning
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    return model, base_model
```

Summary:

CNN with Transfer Learning (Frozen)

Layer (Type)	Output Shape	Parameters
resnet152v2 (Functional)	(32, 5, 5, 2048)	58,331,648
flatten_4 (Flatten)	(32, 51200)	0
dense_8 (Dense)	(32, 128)	6,553,728
dense_9 (Dense)	(32, 1)	129

Total Parameters: 64,885,505 | Trainable: 6,553,857 | Non-trainable: 58,331,648

Total Parameters: 64,885,505 | Trainable: 64,885,505

רשת עמוקה (CNN with Transfer Learning and Fine Tuning) : 2B

בגרסה זו, לאחר אימון ראשוני עם רשת הבסיס הקפואה, בוצע **Fine Tuning** לחלק העליון של הרשת.

שכבות תחתונות נשארו קפואות, ואילו השכבות מעל **layer 540** הוגדרו כ-**Trainable**.

שימוש בקצב למידה נמוך במיוחד (0.0001) אפשר התאמה מדויקת לדאטהסט.

Code:

```python
def create_cnn_with_transfer_learning_finetuned():
    """
    SOLUTION FOR QUESTION 1B (APPROACH 2): Transfer Learning with Fine-tuning
    Report Section: Network 2B - CNN with Transfer Learning (Fine-tuned)
    Creates CNN using pre-trained ResNet152V2 with fine-tuning for domain adaptation.
    ARCHITECTURE DETAILS:
    Base Network:
    - Model: ResNet152V2 pre-trained on ImageNet
    - Input Shape: 160x160x3 RGB images
    - Include Top: False (remove original classification head)
    - Fine-tuning Strategy: Unfreeze last layers (from layer 540 onwards)
    Custom Classification Head:
    - Flatten: Convert ResNet features to 1D vector
    - Dense: 128 neurons, ReLU activation
    - Output: 1 neuron, Sigmoid activation (binary classification)
    FINE-TUNING STRATEGY:
    - Phase 1: Train with frozen base (prevents catastrophic forgetting)
    - Phase 2: Unfreeze top layers and retrain with learning rate
    - Learning Rate: 0.0001
    TRAINING CONFIGURATION:
    - Optimizer: Adam with learning rate (0.0001)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy
    ADVANTAGES:
    - Better adaptation to chest X-ray domain
    - Higher potential accuracy than frozen approach
    - Balances between overfitting and underfitting

    REPORT USAGE: Compare performance with frozen and custom CNN approaches
    """
```

```python
def create_cnn_with_transfer_learning_finetuned():
    # Load pre-trained ResNet152V2
    inputs = tf.keras.Input(shape=(*IMG_SIZE, 3), batch_size=BATCH_SIZE)
    base_model = ResNet152V2(
        weights='imagenet',
        include_top=False,
        input_tensor=inputs
    )

    # Initially freeze base model for stable training
    base_model.trainable = False

    # Create complete model with custom classification head
    model = Sequential([
        base_model,
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    # Compile with low learning rate for initial training
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
```

```python
    # Enable fine-tuning: unfreeze top layers
    base_model.trainable = True
    fine_tune_at = 540  # Unfreeze from this layer onwards

    # Keep early layers frozen to preserve generic features
    for layer in base_model.layers[:fine_tune_at]:
        layer.trainable = False

    # Re-compile with learning rate for fine-tuning
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    return model, base_model
```

Summary:

CNN with Transfer Learning (Fine-tuned)

Layer (Type)	Output Shape	Parameters
resnet152v2 (Functional)	(None, 5, 5, 2048)	**58,331,648**
flatten_5 (Flatten)	(None, 51200)	**0**
dense_10 (Dense)	(None, 128)	**6,553,728**
dense_11 (Dense)	(None, 1)	**129**

Total Parameters: 64,885,505 | Trainable: 15,487,233 | Non-trainable: 49,398,272

Total Parameters: 64,885,505 | Trainable: 64,885,505

משימה 2

1. יש לאמן את שתי הרשתות, ולציין בדוח את אלגוריתם האימון, מספר ה- EPOCHS, RATE LEARNING, גודל BATCH-MINI. לגבי הרשת מבוססת TRANSFER LEARNING, יש להציג שני אופניי אימון: א) פרמטרי רשת הבסיס מוקפאים ורק השכבות שהוספתם מתעדכנות באימון. ב) FINE-TUNING בו מעדכנים גם חלק – לבחירתכם – של שכבות רשת הבסיס. איזה מהשיטות הביאה לביצועים טובים יותר מבחינת ACCURACY? המשיכו בסעיפים הבאים רק עם השיטה הטובה מבין השתיים.
2. עבור שתי הרשתות שתכננתם ציירו את גרף הביצועים , PRECISION - RECALL כאשר כל נקודה בגרף תחושב עבור רמת סף שונה (ביחס להסתברות שמפיקה הרשת) להחלטה על דוגמא חיובית (כלומר עם דלקת ריאות). הסף יהיה בטווח 0.1 עד 0.9 בקפיצות של 0.05 . סמנו גם על הגרף את נקודות SCORE - F שיחושבו מכל זוג ערכים של PRECISION ו- RECALL .
3. עבור איזה סף התקבל ערך SCORE - F הגבוה ביותר?

הסבר: שימו לב שההחלטה האם יש או אין בתמונה דלקת ריאות, מתקבלת ע"י הפעלת סף על ההסתברות שהרשת מוציאה (זכרו שהרשת מפיקה הסתברות רציפה בין 0 ל-1). מוצא הרשת מתקבל באמצעות הפונקציה predict(). במשימה 2 יש לשנות את הסף בתחום 0.1 עד 0.9 (בקפיצות של 0.05) ולחשב את ההחלטות בהתאם לסף: אם ההסתברות במוצא הרשת גבוהה מהסף אז ההחלטה חיובית אחרת ההחלטה שלילית. לדוגמא :נניח שהסף הוא 0.3 והרשת מוציאה הסתברות 0.4, אז היות ש: 0.3<0.4 ההחלטה היא שיש דלקת ריאות. לכל ערך סף תקבלו ביצועים שונים של הרשת, אשר באמצעותם תחשבו צמד אחד של ערכי RECALL-PRECISION. כל נקודה בגרף RECALL-PRECISION תתקבל מערך שונה של הסף.

ייבוא הנתונים והצגה ראשונית

הצגה ראשונית של הדאטהסט

בשלב זה בוצע ייבוא של הדאטהסט הכולל צילומי רנטגן של ריאות תקינות ודלקת ריאות. הנתונים חולקו לשלושה סטים : אימון (Train), אימות (Validation) ובדיקה (Test). לבחינה ראשונית של איכות הנתונים, הוצגו 8 תמונות אקראיות מהסט, דרכן ניתן להבחין בהבדלים בין התמונות התקינות לתמונות עם דלקת.

רשת 1 – CNN ללא Transfer Learning

Code:

```python
def create_cnn_without_transfer_learning():
    """
    SOLUTION FOR QUESTION 1A: Custom CNN Architecture
    Report Section: Network 1 - CNN without Transfer Learning

    Creates a custom CNN model from scratch for binary pneumonia classification.

    ARCHITECTURE DETAILS:
    - Input Layer: 160x160x3 RGB chest X-ray images
    - Convolutional Blocks: 4 identical blocks with:
      * Conv2D: 32 filters, 3x3 kernel, ReLU activation
      * MaxPooling2D: 2x2 pool size, stride=2
    - Feature Extraction: Flatten layer converts 3D to 1D
    - Classification Head:
      * Dense: 128 neurons, ReLU activation
      * Output: 1 neuron, Sigmoid activation (binary classification)

    TRAINING CONFIGURATION:
    - Optimizer: Adam (default learning rate)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy

    REPORT USAGE: Include architecture diagram and parameter count
    """
```

```python
model = Sequential([
    # Explicit input with fixed batch size for clearer diagrams
    tf.keras.Input(shape=(*IMG_SIZE, 3)),
    # First Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Second Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Third Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Fourth Convolutional Block
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    # Feature Extraction
    Flatten(),

    # Classification Head
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification output
])

# Compile model with binary classification settings
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

return model
```

Max Validation Accuracy: 0.9741

רשת 2– Transfer Learning (Frozen)

```python
# ==========================================================================
# QUESTION 1B - CNN WITH TRANSFER LEARNING (FROZEN APPROACH)
# Report Section: Network 2A - Transfer Learning with Frozen Base
# ==========================================================================

Tabnine | Edit | Test | Explain | Document
def create_cnn_with_transfer_learning_frozen():
    """
    SOLUTION FOR QUESTION 1B (APPROACH 1): Transfer Learning with Frozen Base
    Report Section: Network 2A - CNN with Transfer Learning (Frozen)

    Creates CNN using pre-trained ResNet152V2 with frozen weights for feature extraction.

    ARCHITECTURE DETAILS:
    Base Network:
    - Model: ResNet152V2 pre-trained on ImageNet
    - Input Shape: 160x160x3 RGB images
    - Include Top: False (remove original classification head)
    - Trainable: False (frozen weights - no backpropagation)

    Custom Classification Head:
    - Flatten: Convert ResNet features to 1D vector
    - Dense: 128 neurons, ReLU activation
    - Output: 1 neuron, Sigmoid activation (binary classification)

    TRAINING CONFIGURATION:
    - Optimizer: Adam with low learning rate (0.0001)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy

    ADVANTAGES:
    - Fast training (only classification head is trained)
    - Good for small datasets
    - Leverages ImageNet-learned features

    REPORT USAGE: Compare with custom CNN and fine-tuned approach
    """

    # Load pre-trained ResNet152V2 without classification head
    base_model = ResNet152V2(
        weights='imagenet',
        include_top=False,
        input_shape=(*IMG_SIZE, 3)
    )
```

```python
    # Freeze all layers in the base model
    base_model.trainable = False

    # Create complete model with custom classification head
    model = Sequential([
        base_model,
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    # Compile with low learning rate for transfer learning
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    return model, base_model
```

Max Validation Accuracy: 0.9712

רשת 3 – Transfer Learning (Fine-Tuned)

Code:

```python
def create_cnn_with_transfer_learning_finetuned():
    """
    SOLUTION FOR QUESTION 1B (APPROACH 2): Transfer Learning with Fine-tuning
    Report Section: Network 2B - CNN with Transfer Learning (Fine-tuned)
    Creates CNN using pre-trained ResNet152V2 with fine-tuning for domain adaptation.
    ARCHITECTURE DETAILS:
    Base Network:
    - Model: ResNet152V2 pre-trained on ImageNet
    - Input Shape: 160x160x3 RGB images
    - Include Top: False (remove original classification head)
    - Fine-tuning Strategy: Unfreeze last layers (from layer 540 onwards)
    Custom Classification Head:
    - Flatten: Convert ResNet features to 1D vector
    - Dense: 128 neurons, ReLU activation
    - Output: 1 neuron, Sigmoid activation (binary classification)
    FINE-TUNING STRATEGY:
    - Phase 1: Train with frozen base (prevents catastrophic forgetting)
    - Phase 2: Unfreeze top layers and retrain with learning rate
    - Learning Rate: 0.0001
    TRAINING CONFIGURATION:
    - Optimizer: Adam with learning rate (0.0001)
    - Loss Function: Binary Crossentropy
    - Metrics: Accuracy
    ADVANTAGES:
    - Better adaptation to chest X-ray domain
    - Higher potential accuracy than frozen approach
    - Balances between overfitting and underfitting

    REPORT USAGE: Compare performance with frozen and custom CNN approaches
    """
```

```python
def create_cnn_with_transfer_learning_finetuned():
    # Load pre-trained ResNet152V2
    inputs = tf.keras.Input(shape=(*IMG_SIZE, 1), batch_size=BATCH_SIZE)
    base_model = ResNet152V2(
        weights='imagenet',
        include_top=False,
        input_tensor=inputs
    )

    # Initially freeze base model for stable training
    base_model.trainable = False

    # Create complete model with custom classification head
    model = Sequential([
        base_model,
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    # Compile with low learning rate for initial training
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    # Enable fine-tuning: unfreeze top layers
    base_model.trainable = True
    fine_tune_at = 540  # Unfreeze from this layer onwards

    # Keep early layers frozen to preserve generic features
    for layer in base_model.layers[:fine_tune_at]:
        layer.trainable = False

    # Re-compile with learning rate for fine-tuning
    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    return model, base_model
```

Max Validation Accuracy: 0.9597

ניתוח לפי סעיף 1:

בדומה למשימה 1, גם במשימה זו אומנו שלושה מודלים שונים על אותו דאטהסט, במטרה לבדוק איזו גישה מניבה את הביצועים הטובים ביותר.

1.1 - המודלים שאומנו:

- CNN מותאם אישית ללא Transfer Learning (מאפס)
- מודל עם Transfer Learning עם שכבות קפואות בלבד (Frozen)
- מודל עם Transfer Learning עם Fine Tuning (פתיחת שכבות מהשכבה ה־540 ומעלה)

1.2 - פרמטרי האימון (אחידים לכל המודלים):

פרמטר	ערך ברירת מחדל / (No Transfer Frozen)	Fine-Tuned
מספר אפוקים (epochs)	20	20
גודל מיני־באץ' (mini-batch)	32	32
אופטימייזר	Adam	Adam
Learning Rate	0.0001	0.0001

1.3 - הסבר על שני אופני האימון ב־Transfer Learning:

א:Training Frozen .

בגישה זו נעשה שימוש ברשת בסיס (ResNet152V2) שאומנה מראש על ImageNet, כאשר כל שכבות הבסיס הוגדרו כ־ Frozen – כלומר לא התבצע עדכון פרמטרים בשכבות אלו במהלך האימון. רק השכבות שהוספנו בחלק העליון ((Flatten, Dense(128), Dense(1) עודכנו.

ב:Fine Tuning .

בגישה זו, בנוסף לשכבות העליונות, נפתחו לעדכון גם כל השכבות מהשכבה ה־540 ומעלה של רשת הבסיס. כדי למנוע הרס של הידע שנלמד מראש, קצב הלמידה (learning rate) הונמך ל־ 0.0001 שינוי זה אפשר לרשת ללמוד בצורה מדויקת יותר מבלי "לשכוח" את הידע שנרכש באימון המקורי.

1.4 - ממצאים ראשוניים (Accuracy):

הגישה של – Transfer Learning בין אם עם שכבות קפואות ובין אם עם – Fine Tuning מאפשרת למודלים ללמוד תכונות חזותיות עמוקות ומשמעותיות מתוך התמונות, תוך ניצול ידע קודם מרשתות שאומנו על דאטה גדול) כמו.(ImageNet
במונחי דיוק ,(Accuracy) רשת ה־Fine-Tuned השיגה את התוצאה הטובה ביותר, בזכות האפשרות לעדכן חלק מהשכבות העמוקות ברשת הבסיס ולכוונן את הלמידה לדאטה הספציפי של המשימה.

לעומת זאת, המודל שנבנה ללא Transfer Learning נתן ביצועים טובים יחסית באימון, אך קיים חשש ל־ – Overfittingכלומר התאמה יתרה לנתוני האימון והכללה חלשה על דאטה חדש.

לאור זאת, נבחר להמשיך לשלבים הבאים (סעיפים 2–3) עם המודל המבוסס Fine Tuning, שהראה יתרון ברור מבחינת.Accuracy

```python
def evaluate_with_thresholds(model, test_ds, model_name):
    """
    QUESTION 2C - Precision-Recall Analysis with Multiple Thresholds

    Performs comprehensive threshold analysis to find optimal classification thresholds.
    This analysis is crucial for medical diagnosis applications where precision and recall
    balance is critical for clinical decision making.

    Threshold Analysis:
    - Tests thresholds from 0.1 to 0.9 with steps of 0.05
    - Calculates precision, recall, and F1-score for each threshold
    - Identifies optimal threshold based on F1-score maximization
    - Provides detailed performance metrics

    Clinical Relevance:
    - Higher thresholds: Better precision (fewer false positives)
    - Lower thresholds: Better recall (fewer false negatives)
    - Optimal threshold: Best balance for pneumonia detection

    Args:
        model: Trained CNN model
        test_ds: Test dataset
        model_name: Model name for logging

    Returns:
        Dictionary containing:
        - thresholds: Array of tested thresholds
        - precisions: Precision values for each threshold
        - recalls: Recall values for each threshold
        - f1_scores: F1-scores for each threshold
        - best_threshold: Optimal threshold value
        - best_f1: Maximum F1-score achieved
    """
    print(f"\n🔍 Evaluating {model_name} with different thresholds...")

    # Get predictions and true labels
    y_true = []
    y_pred_prob = []

    for images, labels in test_ds:
        y_true.extend(labels.numpy())
        predictions = model.predict(images, verbose=0)
        y_pred_prob.extend(predictions.flatten())

```

```python
    y_true = np.array(y_true)
    y_pred_prob = np.array(y_pred_prob)

    # Calculate precision and recall for different thresholds
    thresholds = np.arange(0.1, 0.95, 0.05)  # From 0.1 to 0.9 with steps of 0.05
    precisions = []
    recalls = []
    f1_scores = []

    print(f"   📊 Testing {len(thresholds)} thresholds: {thresholds}")

    for threshold in thresholds:
        # Apply threshold
        y_pred_binary = (y_pred_prob >= threshold).astype(int)

        # Calculate metrics
        precision = precision_score(y_true, y_pred_binary, zero_division=0)
        recall = recall_score(y_true, y_pred_binary, zero_division=0)
        f1 = f1_score(y_true, y_pred_binary, zero_division=0)

        precisions.append(precision)
        recalls.append(recall)
        f1_scores.append(f1)
      # Find best threshold based on F1-score
    best_f1_idx = np.argmax(f1_scores)
    best_threshold = thresholds[best_f1_idx]
    best_f1 = f1_scores[best_f1_idx]
    best_precision = precisions[best_f1_idx]
    best_recall = recalls[best_f1_idx]

    print(f"\n🏆 Best Results for {model_name}:")
    print(f"   🎯 Best Threshold: {best_threshold:.2f}")
    print(f"   📈 Best F1-Score: {best_f1:.4f}")
    print(f"   📊 Precision at Best F1: {best_precision:.4f}")
    print(f"   📊 Recall at Best F1: {best_recall:.4f}")

    return {
        'thresholds': thresholds,
        'precisions': precisions,
        'recalls': recalls,
        'f1_scores': f1_scores,
        'best_threshold': best_threshold,
        'best_f1': best_f1,
        'best_precision': best_precision,
        'best_recall': best_recall
    }
```

ניתוח לפי סעיף 2:

מטרה:

לבחון את השפעת סף ההחלטה (threshold) על ביצועי המודלים, ולזהות את הסף שמביא ל־F1-Scoreמקסימלי עבור כל אחת מהשיטות.
בשלב זה הערכנו את ביצועי המודלים על סט הבדיקה תוך שימוש במגוון ערכי סף בטווח של 0.1עד 0.9, בקפיצות של 0.05.
עבור כל ערך סף חושבו שלושת המדדים המרכזיים:

- Precision (דיוק) – שיעור התחזיות החיוביות שהיו נכונות.
- Recall (רגישות) – שיעור הדגימות החיוביות שהתגלו על ידי המודל.
- F1-Score - ממוצע הרמוני של דיוק ורגישות, המאזן בין השניים.

דוגמה להסבר המשמעות של הסף:

כאשר לדוגמה הסף מוגדר כ־0.3, וניבוי המודל עבור דוגמה מסוימת הוא 0.4 – היא תסווג כחולה (חיובי).

לעומת זאת, אם הסף היה 0.5, אותה דוגמה תסווג כבריאה (שלילי).

מכאן עולה כי העלאת הסף מעלה את Precision אך מורידה את Recall – ולהיפך.

תוצאות ניתוח F1-Score :

מודל	F1-Scoreמקסימלי	Thresholdאופטימלי
Transfer Learning (Fine-Tuned)	0.8946	0.90
Transfer Learning (Frozen)	0.8907	0.90
No Transfer Learning	0.8584	0.90

הגרפים מראים ששלושת המודלים הגיעו לביצועי השיא שלהם ב- Threshold = 0.9, אך המודל עם Fine Tuningהציג את ערך F1 הגבוה ביותר.

מסקנה:

מודל ה־ **Transfer Learningעם Fine Tuning** הצליח לאזן בצורה הטובה ביותר בין דיוק (Precision)לרגישות – (Recall) גם כאשר סף ההחלטה גבוה יחסית. זהו יתרון משמעותי בהקשרים רפואיים, בהם **זיהוי מוקדם (Recall גבוה** (חשוב במיוחד אך יש צורך גם במניעת **חיוביים שגויים(Precision גבוה)**.

ניתוח לפי סעיף 3:

- על פי ערכי F1 , המודל CNN ללא Transfer Learning הוביל עם התוצאה הגבוהה ביותר : **0.8946** בסף **0.9**.
- כל המודלים הגיעו למקסימום באותו סף, אך Fine-Tuned הצליח לשמור על איזון טוב יותר גם בתנאים קיצוניים.
- תוצאה זו הופקה באמצעות פונקציית create_comparison_summary() שסיכמה את כל ערכי הספים למודלים השונים.
- **מודל מנצח:** Transfer Learning (Fine-tuned) – בזכות דיוק גבוה, רגישות מאוזנת, וגמישות בהתאמה לספים קליניים משתנים.

משימה 3 – בדיקת השפעת אלגוריתם האופטימיזציה והפעלת Early Stopping

משימה 3

1. בדקו את ביצועי הרשתות עם אלגוריתמי האימון הבאים (בדקו השפעת מספר ה EPOCHS וה- LEARNING RATE לכל אלגוריתם):

   א. אלגוריתם SGD
   ב. אלגוריצם SGD עם MOMENTUM.
   ג. אלגוריתם ADAM
   ד. אלגוריתם RMSPROP
   ה. בחרו את האלגוריתם שהביא לתוצאות הטובות ביותר בסעיפים א-ד והפעילו מנגנון EARLY STOPPING בהתאם לאלגוריתם שנלמד בהרצאה (קובץ שקפים בנושא רגולריזציה). האם הושג שיפור בביצועים באמצעות מנגנון זה?

במשימה זאת יש להציג את גרף ההתכנסות של תהליך האימון לכל אחד מהסעיפים כולל TRAIN LOSS ו- VALIDATION LOSS, וכן TRAIN ACCURACY ו- VALIDATION ACCURACY. זאת כאשר נעשה שימוש בסט ולידציה שכולל לפחות 50 תמונות עם דלקת ריאות (25 מכל סוג) ו-50 תמונות ללא דלקת ריאות. סט הולידציה ילקח מתוך סט האימון (כך שיהוו קבוצות זרות של תמונות).

במשימה זו נבחן את השפעת פרמטרי האימון הבאים על ביצועי הרשתות:

- מספר ה- **Epochs**
- ערכי **Learning Rate**
- סוג **ה- Optimizer**

לצורך כך, הרצנו את הקוד על רשת CNN (ללא Transfer Learning) ועל רשת Transfer Learning (Fine-Tuned), תוך בדיקת 4 Optimizers שונים:

1. SGD
2. SGD עם Momentum
3. Adam
4. RMSprop

השתמשנו בערכי Learning Rate: 0.01, 0.001, 0.0001
השתמשנו בערכי Epochs: 5, 10, 15 , 17.

נוצרו גרפים עבור כל אופטימיזר, כאשר כל גרף מציג:

- Train Loss
- Validation Loss
- Train Accuracy
- Validation Accuracy

1) השפעת Optimizer, Learning Rate ו- Epochs

רשת CNN ללא Transfer learning

א. אלגוריתם SGD

בדקנו את השפעת SGD הקלאסי ללא Momentum.

- **SGD - Graph for learning rate 0.01:**
- **Epochs 5:**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9566, Validation Accuracy: 0.9530
- Best Validation Accuracy: 0.9530 (Epoch 8)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.9650, Validation Accuracy: 0.9511
- Best Validation Accuracy: 0.9684 (Epoch 12)
- Total Images: 1043 training, 1043 validation

- **SGD - Graph for learning rate 0.001:**

❖ **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 1)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 1)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

- **SGD - Graph for learning rate 0.0001:**

❖ **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 2)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

ב. אלגוריתם SGD עם Momentum

נוספה תמיכה בפרמטר Momentum = 0.9

- **SGD with Momentum - Graph for learning rate 0.01:**

❖ **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9703, Validation Accuracy: 0.9549**
- **Best Validation Accuracy: 0.9655 (Epoch 4)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9756, Validation Accuracy: 0.9722
- Best Validation Accuracy: 0.9722 (Epoch 10)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- **Final Training Accuracy: 0.9844, Validation Accuracy: 0.9703**
- **Best Validation Accuracy: 0.9703 (Epoch 9)**
- **Total Images: 1043 training, 1043 validation**

- **SGD with Momentum - Graph for learning rate 0.001:**

❖ **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9274, Validation Accuracy: 0.9473**
- **Best Validation Accuracy: 0.9473 (Epoch 5)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9597, Validation Accuracy: 0.9377
- Best Validation Accuracy: 0.9578 (Epoch 9)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.9633, Validation Accuracy: 0.9674
- Best Validation Accuracy: 0.9674 (Epoch 15)
- Total Images: 1043 training, 1043 validation

- **SGD with Momentum - Graph for learning rate 0.0001:**

- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 1)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 1)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 15:**

Summary Metrics:
- **Final Training Accuracy: 0.8584, Validation Accuracy: 0.8945**
- **Best Validation Accuracy: 0.8945 (Epoch 15)**
- **Total Images: 1043 training, 1043 validation**

ג. אלגוריתם Adam

אחד האופטימייזרים הפופולריים ללמידה עמוקה – משלב Adaptive Learning ו-Momentum.

- **Adam - Graph for learning rate 0.01:**
- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9408, Validation Accuracy: 0.9406**
- **Best Validation Accuracy: 0.9406 (Epoch 5)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.9844, Validation Accuracy: 0.9664
- Best Validation Accuracy: 0.9703 (Epoch 12)
- Total Images: 1043 training, 1043 validation

- **Adam - Graph for learning rate 0.001:**

- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9703, Validation Accuracy: 0.9396**
- **Best Validation Accuracy: 0.9751 (Epoch 4)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- **Final Training Accuracy: 0.9854, Validation Accuracy: 0.9751**
- **Best Validation Accuracy: 0.9770 (Epoch 9)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.9938, Validation Accuracy: 0.9770
- Best Validation Accuracy: 0.9847 (Epoch 12)
- Total Images: 1043 training, 1043 validation

- **Adam - Graph for learning rate 0.0001:**

- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9600, Validation Accuracy: 0.9444**
- **Best Validation Accuracy: 0.9521 (Epoch 3)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9693, Validation Accuracy: 0.9693
- Best Validation Accuracy: 0.9693 (Epoch 10)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- **Final Training Accuracy: 0.9715, Validation Accuracy: 0.9588**
- **Best Validation Accuracy: 0.9722 (Epoch 13)**
- **Total Images: 1043 training, 1043 validation**

ד. אלגוריתם RMSprop

אופטימייזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

- **RMSprop - Graph for learning rate 0.01:**
- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9080, Validation Accuracy: 0.9511**
- **Best Validation Accuracy: 0.9511 (Epoch 5)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9595, Validation Accuracy: 0.9626
- Best Validation Accuracy: 0.9626 (Epoch 10)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- **Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613**
- **Best Validation Accuracy: 0.7613 (Epoch 1)**
- **Total Images: 1043 training, 1043 validation**

- **RMSprop - Graph for learning rate 0.001:**

- **Epochs 5:**

Summary Metrics:
- Final Training Accuracy: 0.9612, Validation Accuracy: 0.9003
- Best Validation Accuracy: 0.9626 (Epoch 3)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9861, Validation Accuracy: 0.9511
- Best Validation Accuracy: 0.9808 (Epoch 7)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- **Final Training Accuracy: 0.9904, Validation Accuracy: 0.9770**
- **Best Validation Accuracy: 0.9770 (Epoch 15)**
- **Total Images: 1043 training, 1043 validation**

- **RMSprop - Graph for learning rate 0.0001:**

- **Epochs 5:**

Summary Metrics:
- **Final Training Accuracy: 0.9329, Validation Accuracy: 0.8715**
- **Best Validation Accuracy: 0.9338 (Epoch 3)**
- **Total Images: 1043 training, 1043 validation**

❖ **Epochs 10:**

Summary Metrics:
- Final Training Accuracy: 0.9561, Validation Accuracy: 0.9578
- Best Validation Accuracy: 0.9578 (Epoch 10)
- Total Images: 1043 training, 1043 validation

❖ **Epochs 15:**

Summary Metrics:
- Final Training Accuracy: 0.9686, Validation Accuracy: 0.9664
- Best Validation Accuracy: 0.9664 (Epoch 15)
- Total Images: 1043 training, 1043 validation

רשת CNN עם Fine Tuned Transfer learning

א. אלגוריתם SGD

בדקנו את השפעת SGD הקלאסי ללא Momentum.

- **SGD - Graph for learning rate 0.01:**
- **Epochs 17:**

Test accuracy = 89.16%

- **SGD - Graph for learning rate 0.001:**
- **Epochs 17:**

- Test accuracy = 93.33%

- **SGD - Graph for learning rate 0.0001:**
- **Epochs 17:**

Test accuracy = 72.66%

ב. אלגוריתם SGD עם Momentum

נוספה תמיכה בפרמטר Momentum = 0.9

- **SGD with Momentum - Graph for learning rate 0.01:**
- **Epochs 17:**

Test accuracy = 84.83%

- **SGD with Momentum - Graph for learning rate 0.001:**
- **Epochs 17:**

- Test accuracy = 56.66%

- **SGD with Momentum - Graph for learning rate 0.0001:**
  - **Epochs 17:**

Test accuracy = 42.33%

ג. אלגוריתם Adam

אחד האופטימייזרים הפופולריים ללמידה עמוקה – משלב Adaptive Learning ו-Momentum.

- **Adam - Graph for learning rate 0.01:**
- **Epochs 17:**

Test accuracy = 92.66%

- **Adam - Graph for learning rate 0.001:**
- **Epochs 17:**

- Test accuracy = 92.76%

- **Adam - Graph for learning rate 0.0001:**
- **Epochs 17:**

Test accuracy = 91.16%

ד. אלגוריתם RMSprop

אופטימייזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

- **RMSprop - Graph for learning rate 0.01:**
- **Epochs 17:**

Test accuracy = 90.83%

- **RMSprop - Graph for learning rate 0.001:**
- **Epochs 17:**

➢

➢

➢ Test accuracy = 89.95%

➢ **RMSprop - Graph for learning rate 0.0001:**

❖ **Epochs 17:**

- Test accuracy = 91.00%

ה. הפעלת Early Stopping

בהתבסס על השוואת התוצאות בין האופטימייזרים בסעיפים א'-ד', נמצא במודל CNN בסיסי ללא Transfer Learning האלגוריתם עם **Adam** עם **Learning Rate = 0.001** וב- **15 Epochs** הביא לביצועים הטובים ביותר, עם דיוק אימות של **0.9770** ודיוק אימון של **0.9938**, תוך התכנסות יציבה ומהירה.

לכן, לצורך סעיף זה, הופעל מנגנון **EarlyStopping** על המודל CNN בסיסי ללא Transfer Learning עם Adam , תוך שימוש בפרמטרים הבאים:

- patience=5
- ניטור אחר val_loss
- restore_best_weights=True
- הגבלת למקסימום של 50 אפוקים (אך עצירה מוקדמת צפויה)

תוצאה:

- האימון נעצר לאחר **11 אפוקים בלבד**
- Accuracyסופי: **0.9760**
- Validation Accuracyמיטבי: **0.9760** (ב- Epoch 11)
- Validation Lossמיטבי: **0.0755** (ב- Epoch 8)
- חסכון בזמן אימון: כ- **63.3%** יחסית להרצה מלאה של 30 אפוקים

שיפור בביצועים:

למרות ירידה זניחה של כ- **1.4%** בדיוק האימות בהשוואה להרצה ללא עצירה מוקדמת, הושג חסכון ניכר בזמן האימון ומניעת – Overfitting כלומר, התקבל איזון טוב יותר בין דיוק לבין יעילות. לכן, ניתן לקבוע שהשימוש ב- EarlyStopping שיפר את ההתנהגות הכוללת של המודל בתנאי אימון מציאותיים.

Early Stopping Summary:
- Stopped at epoch 11 (requested max: 30)
- Final accuracy: 0.9760 (train: 0.9842)
- Best validation accuracy: 0.9760 at epoch 11
- Best validation loss: 0.0755 at epoch 8
- Total Images: 1043 training, 1043 validation
- Training time savings: 63.3%

Early Stopping Impact Analysis for Adam (LR=0.001)
Accuracy Comparison: Regular Training vs. Early Stopping
Final: 0.9770
Final: 0.9636
Accuracy
Epoch
Regular Training Accuracy
Regular Validation Accuracy
Early Stopping Training Accuracy
Early Stopping Validation Accuracy
Loss Comparison: Regular Training vs. Early Stopping
Final: 0.1150
Final: 0.1027
Loss
Epoch
Regular Training Loss
Regular Validation Loss
Early Stopping Training Loss
Early Stopping Validation Loss
Early Stopping vs Regular Training Comparison:
• Regular Training: 15 epochs, final val_acc: 0.9770, best: 0.9847
• Early Stopping: 8 epochs, final val_acc: 0.9636, best: 0.9741
• Accuracy change: -0.0134 (-1.4%)
• Training time saved: 73.3% (22 of 30 epochs)

משימה 4 – סיווג מרובה קטגוריות באמצעות CNN

<u>משימה 4</u>

ממשו באמצעות שינוי הרשת ללא TRANSFER LEARNING (ממשימה 1) פתרון לסיווג 3 קטגוריות: (1) אין דלקת (2) יש דלקת חיידקית (3) יש דלקת נגיפית. שימו לב שיש להשתמש במקרה זה ב-SOFTMAX במוצא הרשת. חזרו על משימה 3 עבור הרשת במשימה זאת, והציגו עבור הפתרון הטוב ביותר שהשגתם את ה-CONFUSION MATRIX על סט הבדיקה.

במשימה זו נדרשנו להתאים את רשת ה- CNN מהמשימה הראשונה (ללא Transfer Learning) לסיווג של שלוש קטגוריות:

- אין דלקת (Normal)
- דלקת חיידקית (Bacterial Pneumonia)
- דלקת נגיפית (Viral Pneumonia)

לצורך כך, שונתה שכבת הפלט לשכבת Dense עם שלושה נוירונים ופונקציית אקטיבציה מסוג Softmax, כדי לאפשר סיווג מרובה קטגוריות.

אימון הרשת עם אופטימייזרים שונים

בדומה למשימה 3, בוצעו 36 ריצות אימון של המודל – עבור כל שילוב של:

- שלושה ערכי Learning Rate: 0.01, 0.001, 0.0001
- שלושה ערכי Epochs: 5, 10, 15
- ארבעה אופטימייזרים:
  1. SGD
  2. SGD עם Momentum
  3. Adam
  4. RMSprop

הקוד יצר 36 גרפים המציגים את תהליך האימון, כולל:

- Train Accuracy
- Validation Accuracy
- Train Loss
- Validation Loss

א. אלגוריתם SGD

בדקנו את השפעת SGD הקלאסי ללא Momentum.

- **SGD - Graph for learning rate 0.01:**

❖ Epochs 5:

Max Validation Accuracy: 0.6511

❖ Epochs 10:

Max Validation Accuracy: 0.7445

❖ Epochs 15:

Max Validation Accuracy: 0.7555

- **SGD - Graph for learning rate 0.001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.4908

❖ **Epochs 10:**

Max Validation Accuracy: 0.4908

❖ **Epochs 15:**

Max Validation Accuracy: 0.6044

➢ **SGD - Graph for learning rate 0.0001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.4908

❖ Epochs 10:

Max Validation Accuracy: 0.4908

❖ Epochs 15:

Max Validation Accuracy: 0.4908

ב. אלגוריתם SGD עם Momentum

נוספה תמיכה בפרמטר Momentum = 0.9

- **SGD with Momentum - Graph for learning rate 0.01:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7555

❖ **Epochs 10:**

Max Validation Accuracy: 0.7875

❖ **Epochs 15:**

Max Validation Accuracy: 0.8022

➢ **SGD with Momentum - Graph for learning rate 0.001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7436

❖ Epochs 10:

Max Validation Accuracy: 0.7546

❖ Epochs 15:

Max Validation Accuracy: 0.7683

- **SGD with Momentum - Graph for learning rate 0.0001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.4908

❖ **Epochs 10:**

Max Validation Accuracy: 0.6007

❖ **Epochs 15:**

Max Validation Accuracy: 0.4908

ג. אלגוריתם Adam

אחד האופטימייזרים הפופולריים ללמידה עמוקה – משלב Adaptive Learning ו-Momentum.

➢ **Adam - Graph for learning rate 0.01:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7674

❖ Epochs 10:

Max Validation Accuracy: 0.7720

❖ Epochs 15:

Max Validation Accuracy: 0.7582

- **Adam - Graph for learning rate 0.001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7885

❖ **Epochs 10:**

Max Validation Accuracy: 0.7985

❖ **Epochs 15:**

Max Validation Accuracy: 0.8022

➢ **Adam - Graph for learning rate 0.0001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7756

❖ Epochs 10:

Max Validation Accuracy: 0.7875

❖ Epochs 15:

Max Validation Accuracy: 0.7894

ד. אלגוריתם RMSprop

אופטימייזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

- **RMSprop - Graph for learning rate 0.01:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.4908

❖ **Epochs 10:**

Max Validation Accuracy: 0.7546

❖ **Epochs 15:**

➢ **RMSprop - Graph for learning rate 0.001:**

❖ **Epochs 5:**

❖ Epochs 10:

Max Validation Accuracy: 0.8004

❖ Epochs 15:

Max Validation Accuracy: 0.8114

- **RMSprop - Graph for learning rate 0.0001:**

❖ **Epochs 5:**

Max Validation Accuracy: 0.7509

❖ **Epochs 10:**

Max Validation Accuracy: 0.7756

❖ **Epochs 15:**

Max Validation Accuracy: 0.7802

סיכום השוואתי בין כל הקונפיגורציות

לאחר סיום האימונים, בוצעה השוואה בין המודלים לפי דיוק על סט הוולידציה.
ניתן לראות בגרף הסיכום כי הקונפיגורציה המובילה הייתה:

- Optimizer: RMSprop
- Learning Rate: 0.001
- Epochs: 15
- Validation Accuracy: 0.8114

המודל הזה נבחר כמודל המנצח למשך שימוש ב- Confusion Matrix בסעיף הבא.

```
================================================================================
EXPERIMENT RESULTS SUMMARY
================================================================================

Rank   Experiment                          Optimizer       LR        Epochs   Max Val Acc
-----------------------------------------------------------------------------------------------
1      Task4_RMSprop_LR0.001_Epochs15      RMSprop         0.0010    15       0.8114          👈 BEST
2      Task4_SGD_Momentum_LR0.01_Epochs15  SGD_Momentum    0.0100    15       0.8022
3      Task4_Adam_LR0.001_Epochs15         Adam            0.0010    15       0.8022
4      Task4_RMSprop_LR0.001_Epochs10      RMSprop         0.0010    10       0.8004
5      Task4_Adam_LR0.001_Epochs10         Adam            0.0010    10       0.7985
6      Task4_RMSprop_LR0.001_Epochs5       RMSprop         0.0010    5        0.7949
7      Task4_Adam_LR0.0001_Epochs15        Adam            0.0001    15       0.7894
8      Task4_Adam_LR0.001_Epochs5          Adam            0.0010    5        0.7885
9      Task4_SGD_Momentum_LR0.01_Epochs10  SGD_Momentum    0.0100    10       0.7875
10     Task4_Adam_LR0.0001_Epochs10        Adam            0.0001    10       0.7875

================================================================================================
🏆 BEST OVERALL MODEL: Task4_RMSprop_LR0.001_Epochs15
   Max Validation Accuracy: 0.8114 (81.14%)
================================================================================================
```

Confusion Matrix לסיווג תלת־קטגורי

במשימה 4 אנו מסווגים את התמונות לשלוש קטגוריות נפרדות Normal, Bacterial ו-Viral. תמונות ה- Pneumonia משויכות באופן דטרמיניסטי לתת-הקטגוריות Bacterial או Viral על בסיס מילות מפתח בשם הקובץ, כך שמתקבלת תווית תלת-קטגורית עקבית בכל פיצולי הנתונים. מטריצת הבלבול להלן מציגה את הביצועים על פני שלוש הקטגוריות הללו.

Accuracy : 81.14%

הדיוק הגבוה בתאי האלכסון (במיוחד ב- Bacterial) והפיזור הנמוך בתאי הטעות מצביעים על יכולת למידה חזקה של הרשת, גם בהקשר של סיווג מרובה קטגוריות.

סיכום של משימה 4

- כל מודל אומן מהתחלה (לא בוצע Transfer Learning) עם שינוי בשכבת הפלט ל- Softmax.
- המודל הטוב ביותר הושג באמצעות RMSprop עם קצב למידה 0.001 ו- 15 אפוקים.
- בוצעה סימולציה מוצלחת של Confusion Matrix ל- 3 קטגוריות.
- כל הקונפיגורציות תועדו באמצעות גרפים בהתאם לדרישות.

152/200 Images_Normal = 76%

61/100 Images_Bac = 61%

85/100 Images_Viral = 85%

מסקנה: גם ללא Transfer Learning, ניתן להשיג תוצאות טובות בסיווג מרובה קטגוריות כאשר בוחרים בקפידה את הפרמטרים ומבצעים השוואה שיטתית.

סיכום כללי – דוח פרויקט: גילוי דלקת ריאות בצילומי רנטגן

במהלך הפרויקט פיתחנו והערכנו מודלים שונים מבוססי CNN לגילוי דלקת ריאות מסוגים שונים בצילומי רנטגן, תוך השוואה בין שיטות למידה עמוקה עם וללא Transfer Learning , בחינת אופטימיזרים שונים, השפעת פרמטרים היפר־פרמטריים, והרחבת הבעיה לסיווג רב-קטגורי.

מהלכי העבודה המשותפים לכל המשימות

בכל המשימות הקפדנו לבצע חלוקה אחידה בין סט האימון, האימות והבדיקה – כך שנשמרה עקביות והימנעות מ- Data Leakage. בנוסף, בוצעה הדמיה ויזואלית של נתונים לצורך בדיקה ראשונית, ונמדדו מדדים עקביים כגון Accuracy, Precision, Recall ו- F1. כל רשת אומנה לאורך טווחים אחידים של Epochs ו- Learning Rate, והתקננו את תהליך ההשוואה בעזרת גרפים תואמים.

שיאי הביצועים (Highlights)

- המודל עם Transfer Learning ו- Fine Tuning (משימה 2) הציג את ה- F1-Score הגבוה ביותר (**0.8946**) ו- Validation Accuracy גבוהה, מה שהעיד על התאמה טובה ומאוזנת.
- ב־משימה 3 נמצא כי האופטימייזר **Adam** עם learning rate=0.001 הניב את התוצאות הטובות ביותר, ולאחר הפעלת **EarlyStopping** התקבל חיסכון של 63.3% בזמן האימון ללא פגיעה משמעותית בדיוק.
- ב־משימה 4, לאחר התאמת הרשת לסיווג 3 קטגוריות והפעלה חוזרת של ניסויים כמו במשימה 3, נמצא שהקונפיגורציה **RMSprop + learning rate=0.001 + 15 Epochs** סיפקה את הביצועים הטובים ביותר, עם דיוק ולידציה של **0.8114** ו- Confusion Matrix מובהק.

מסקנות מרכזיות

- **Transfer Learning** מתאים במיוחד כאשר יש מגבלה בכמות הדאטה, אך דורש Fine Tuning מדויק כדי להימנע מ- Underfitting או Overfitting.
- **בחירת האופטימייזר** היא קריטית Adam – ורמות למידה מתונות הציגו תוצאות מובהקות ויציבות.
- **EarlyStopping** הוא כלי יעיל למניעת Overfitting ולייעול זמן האימון – במיוחד כשהוא משולב עם מנגנוני למידה מתקדמים.
- **סיווג מרובה קטגוריות** (Normal/Bacterial/Viral) אפשרי גם ללא Transfer Learning, כאשר נעשה שימוש נכון בשכבת Softmax ואופטימיזציה מותאמת.