



המחלקה להנדסת חשמל ואלקטרוניקה

מערכות לומדות ולמידה عمוקה
סמסטר ב – תשפ"ה 2025

מטלת סוף

משימת הפרויקט: גילוי דלקת ריאות בצלומי רנטגן



בשarra חביב

פרנסיס עבוד

מריה נחלה

שם המרצה: פרופ' אמיר אדר

תאריך: 23/08/2025

תוכן העניינים

2	1) תוכן העניינים
4	2) מבוא - הגדרת העבודה
5	3) משימה 1 – תכנון רשות CNN
5	הדמיית תמונות מהדעתה
6	רשות 1 – ללא Transfer Learning (רשות מותאמת אישית)
	Error! Bookmark not defined.....(CNN with Transfer Learning) :
11	רשות عمוקה (CNN with Transfer Learning and Fine Tuning)
4	4) משימה 2 – אימון והערכת ביצועים של מודלים
13	יבוא הנתונים והציג ראיונות
	Error! Bookmark not
	Fine Tuning – Transfer Learning – בעומת שיטות אימון מודלים והשוואת שיטות
14	רשות 1 – ללא Transfer Learning
15	רשות 2 – Transfer Learning (Frozen)
16	רשות 3 – Transfer Learning (Fine-Tuned)
	Error! Bookmark not defined.....Precision-Recall
	Error! Bookmark not defined.....בחרת מודל מנכח לפי F1
21	5) משימה 3 – בדיקת השפעת אלגוריתם האופטימיזציה והפעלת Epochs – Optimizer, Learning Rate
22	השפעת Epochs
22	א. אלגוריתם SGD
22	SGD - Graph for learning rate 0.01
25	SGD - Graph for learning rate 0.001
28	SGD - Graph for learning rate 0.0001
31	ב. אלגוריתם SGD עם Momentum
31	SGD with Momentum - Graph for learning rate 0.01
34	SGD with Momentum - Graph for learning rate 0.001
37	SGD with Momentum - Graph for learning rate 0.0001
40	ג. אלגוריתם Adam
40	Adam - Graph for learning rate 0.01
43	Adam - Graph for learning rate 0.001
46	Adam - Graph for learning rate 0.0001
49	ד. אלגוריתם RMSprop
49	RMSprop - Graph for learning rate 0.01

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

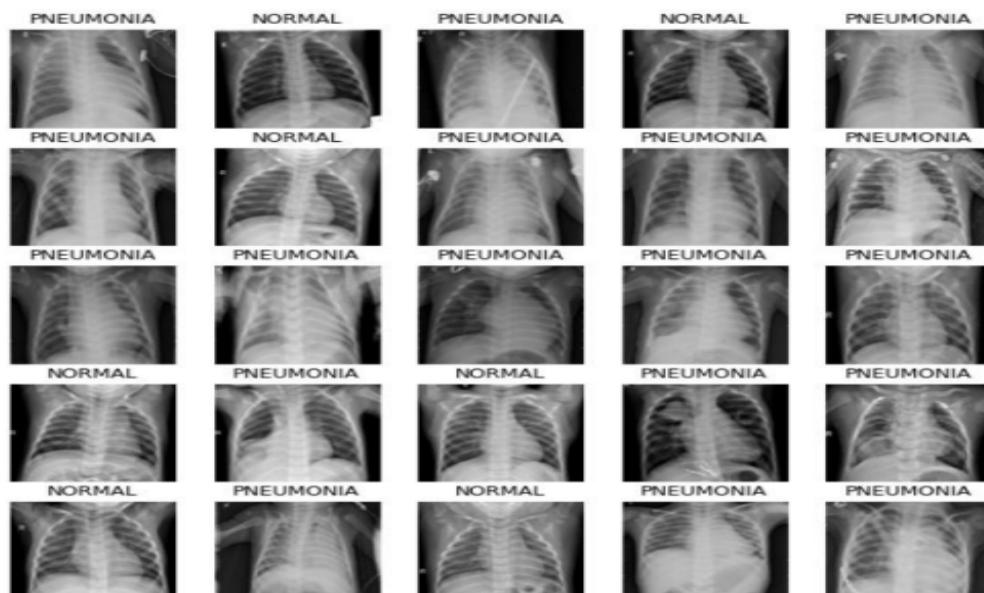
משימה 2 – אימון והערכת ביצועים של מודלים

מבוא - הגדרת העבודה

בפרויקט ימומשו פתרונות מבוססי CNN לגילוי דלקת ריאות מ- 5863 צילומים אמיתיים הכוללים אבחון של רפואיים. 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) וונגיפית (Viral), להלן דוגמאות:



מטרת הפרויקט היא תכנון רשותות למידה عمוקה לגילוי דלקת ריאות, כאשר הביצועים ימדדו על סט תמונות בדיקה, אשר יופרד באופן מוחלט מסט האימון. ביצועים טובים יחשבו לסיכון שגיאה נמוך מ- 7% (כלומר Accuracy גבוהה 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 סעיפים הפרויקט.

משימה 1 – תכנון רשתות CNN

עליכם לתכנן שתי רשתות ע深深ות שונות לפתרון הבעיה: כניסה כל רשת عمוקה היא תמונה והמוצא הינו ההסתברות שהתמונה מייצגת מקרה חובי של דלקת ריאות (בטעות זה אין זה משנה מה סוג הדלקת - חידקית 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 שנייה: רשת CNN מבוססת TRANSFER LEARNING

МОУНОР להיעזר בהצעות לפתרונות המופיעים ב קישור הבא (מדובר על הצעות של אנשים פרטיים ולא פתרון "רשמי"):

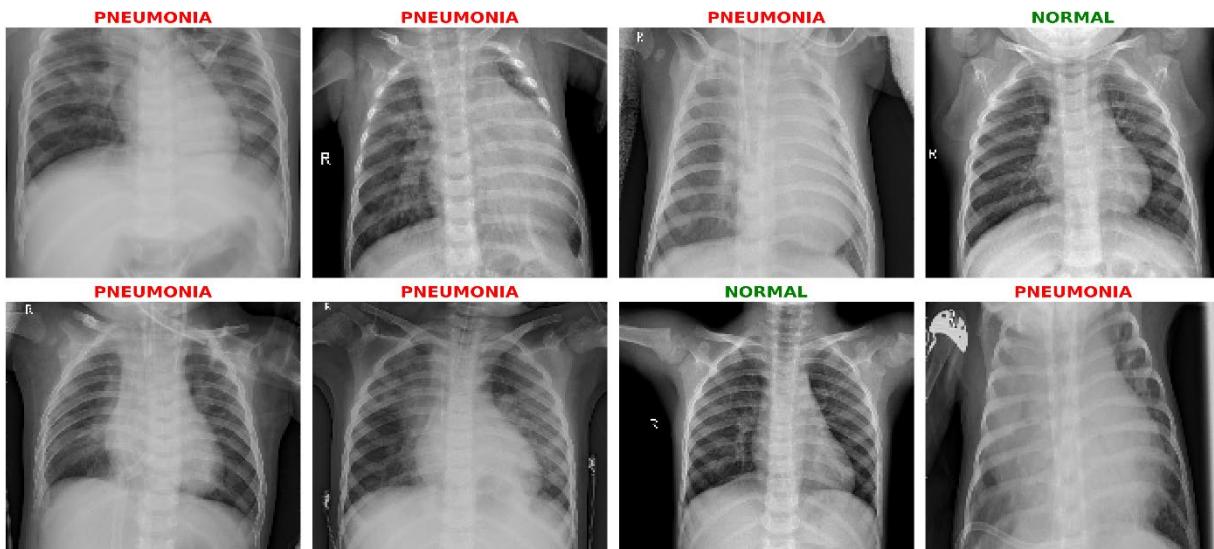
<https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia/code>

יש לפרט בדוח הפרויקט 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 הוספות להשלמת הפתרון המדרש.

הדמיית תמונות מהDataset

לפני בניית הרשתות, ביצעו הדמייה של 8 תמונות לדוגמה מהDataset. ניתן להבחין בהבדל בין צילומים תקינים לצילומים עם דלקת ריאות.

Dataset Sample Images - Chest X-Ray Pneumonia Detection



[Dataset: NORMAL, PNEUMONIA • Image Size: 160×160 • Batch Size: 32]

רשות 1 – ללא Transfer Learning (רשות מותאמת אישית)

רשות זו נבנתה באופן ידני ולא שימוש במודלים מוכנים) כגון ResNet או, (VGG קלומר כל שכבותיה הוגדרו מאפס (from scratch). הרשות כוללת ארבעה בלוקים זהים של שכבות Convolution ו MaxPooling, שכל אחד מהם מיועד לחץ תכונות רלוונטיות מהתמונה ולהפחית את ממדיה באופן הדרגתי. מטרת המבנה זהה היא לאפשר לרשות ללמידה ייצוגים היררכיים – מקוונים פשוטים ועד אזוריים מורכבים יותר – לפני המעבר לשלב הסיוג הסופי. הרשות פותחה עם מטרת סיוג בינארי (דלקת ריאות לעומת מצב תקין), תוך שימוש בשכבת פלט אחת עם פונקציית אקטיבציה מסווג Sigmoid. פונקציית האיבוד שנבחרה היא Binary Crossentropy ואופטימיזר מסווג Adam שמש לעדכון המשקלים במהלך האימון.

הרכיב השכבות:

: Rescaling (1)

משנה את ערך של הפיקסלים בתמונה מ-[0-255] ל-[0-1]

:Conv2D (2)

שכבה שיכולה לשמש כשכבה עליונה ברשות, כפי שהסביר קודם, השכבה לוקחת את התמונה הדו ממדית ומבצעת פעולות כמו סינון, זיהוי קצוטות וכו'.

בחירת גדים :

- מספר מסננים 32
- גודל ה- KERNEL שווה ל- 3X3
- פונקציית האקטיבציה של השכבה "Relu"

: Maxpooling2D (3)

תפקיד השכבה הוא להקטין את עלות החישוב (מקטינה את ממדיה המטሪיצה הדו ממדית שנכנסת לשכבה).

בחירת גדים :

- גודל חלון : 2×2 (קלומר 2 = Strides)

: Flatten (4)

משתמשים בשכבה זו לייצרת פלט שטוח. המטרה משכבה זו הוא להפוך מטሪיצה דו ממדית של תכונות לוktor שנייתן להכניסה אותו לשכבה CNN Classifier.

: Dense (5)

שכבה שמייצגת כפל וקטורים במטሪיצה. הערכים במטሪיצה הם הפרמטרים הניתנים לאימון המתעדכנים במהלך ההפצה.

- שכבה ראשונה עם 128 נוירונים (ReLU)
- שכבת פלט עם נוירון 1 (Sigmoid)

Code:

```

143 def create_cnn_without_transfer_learning():
144     """
145         SOLUTION FOR QUESTION 1A: Custom CNN Architecture
146         Report Section: Network 1 - CNN without Transfer Learning
147
148         Creates a custom CNN model from scratch for binary pneumonia classification.
149
150         ARCHITECTURE DETAILS:
151         - Input Layer: 160x160x3 RGB chest X-ray images
152         - Convolutional Blocks: 4 identical blocks with:
153             * Conv2D: 32 filters, 3x3 kernel, ReLU activation
154             * MaxPooling2D: 2x2 pool size, stride=2
155         - Feature Extraction: Flatten layer converts 3D to 1D
156         - Classification Head:
157             * Dense: 128 neurons, ReLU activation
158             * Output: 1 neuron, Sigmoid activation (binary classification)
159
160         TRAINING CONFIGURATION:
161         - Optimizer: Adam (default learning rate)
162         - Loss Function: Binary Crossentropy
163         - Metrics: Accuracy
164
165         REPORT USAGE: Include architecture diagram and parameter count
166         """

```

```

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

```

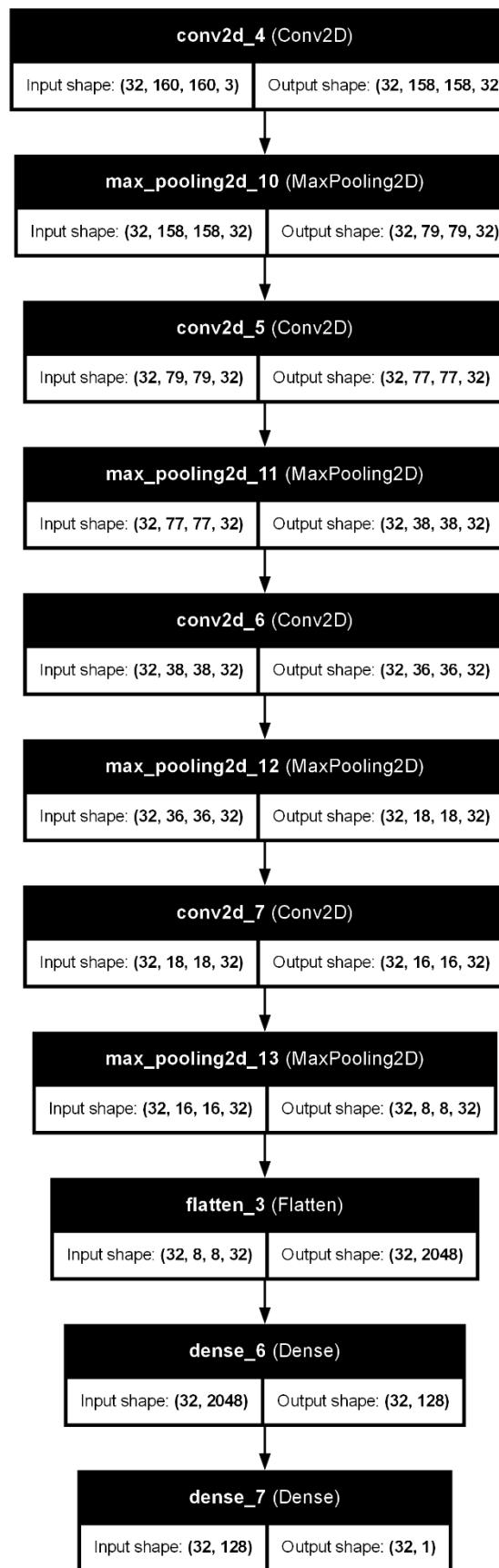
Summary:

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



CNN with Transfer Learning :

רשת זו עשויה שימוש ב-Transfer Learning על בסיס **ResNet152V2** – מודל عمוק שאותן מראש על מאגר התמונות. הרשת נבחרה בזכות הדיק הגדולה שלה ומשום יכולתה ללמידה ייצוגים חזותיים מורכבים ורב-שבביים. השתמשנו **בכל השכבות הקודומות של המודל (כולל כובדיהם) אך השטנו את ה-Top** (שכבת הסיווג המקורי), שכן מטרתנו שונה מזו של **ImageNet** (סיווג ל-1000 קטגוריות) כדי שנוכל להוסיף סיווג מותאם לבעה שלנו ואנו מבצעים סיווג ביןארי.

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

1. **Flatten** – הופכת את המטריצה התלת-ממדית של הפלט מהמודל לקלט וקטורי רצף לשכבות הסיווג.
 2. **Dense (ReLU, נוירונים 128)** – שכבה נוספת הלומדת תכונות לא ליניאריות ומסיימת בהתאם לבעה הספציפית.
 3. **Sigmoid (Dense 1, נוירון)** – שכבת הפלט שמבצעת סיווג ביןארי (דלקת ריאות/בריא) עם הסתברות כעריך בין 0 ל-1.
- חשוב לציין של שכבות ה-ResNet הוגדרו כ-**Frozen** כדי לשמור את הידע המקורי, ומונענו בכך אימון מחדש שעלול להוביל ל-**Overfitting** על דатаה קטו.

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:

```

210 def create_cnn_with_transfer_learning_frozen():
211     SOLUTION FOR QUESTION 1B (APPROACH 1): Transfer Learning with Frozen Base
212     Report Section: Network 2A - CNN with Transfer Learning (Frozen)
213
214     Creates CNN using pre-trained ResNet152V2 with frozen weights for feature extraction.
215
216     ARCHITECTURE DETAILS:
217     Base Network:
218         - Model: ResNet152V2 pre-trained on ImageNet
219         - Input Shape: 160x160x3 RGB images
220         - Include Top: False (remove original classification head)
221         - Trainable: False (frozen weights - no backpropagation)
222
223     Custom Classification Head:
224         - Flatten: Convert ResNet features to 1D vector
225         - Dense: 128 neurons, ReLU activation
226         - Output: 1 neuron, Sigmoid activation (binary classification)
227
228     TRAINING CONFIGURATION:
229         - Optimizer: Adam with low learning rate (0.0001)
230         - Loss Function: Binary Crossentropy
231         - Metrics: Accuracy
232
233     ADVANTAGES:
234         - Fast training (only classification head is trained)
235         - Good for small datasets
236         - Leverages ImageNet-learned features
237
238     REPORT USAGE: Compare with custom CNN and fine-tuned approach
239 
```

```

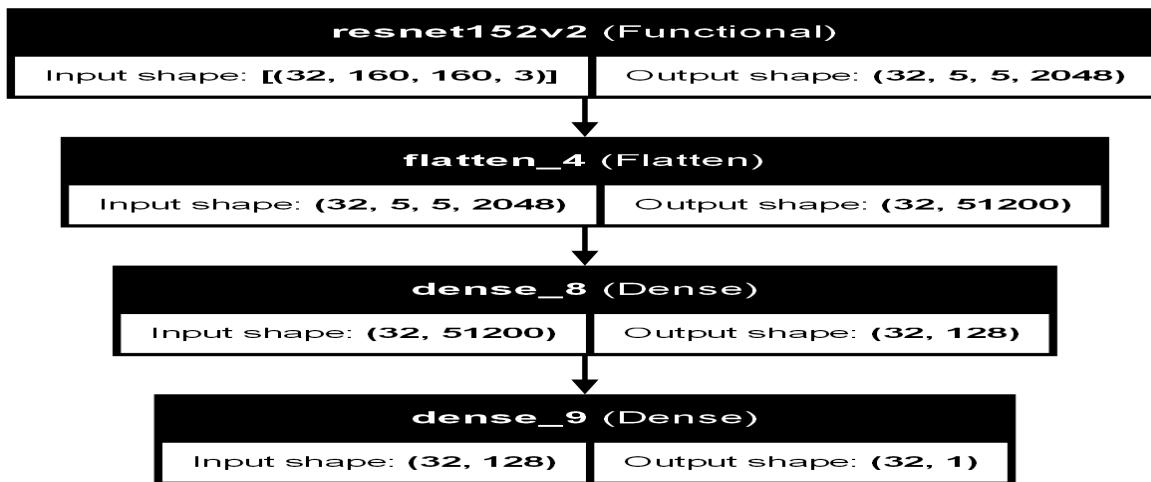
210 v def create_cnn_with_transfer_learning_frozen():
211     # Load pre-trained ResNet152V2 without classification head
212     inputs = tf.keras.Input(shape=(*IMG_SIZE, 3), batch_size=BATCH_SIZE)
213     base_model = ResNet152V2(
214         weights='imagenet',
215         include_top=False,
216         input_tensor=inputs
217     )
218
219     # Freeze all layers in the base model
220     base_model.trainable = False
221
222     # Create complete model with custom classification head
223     model = Sequential([
224         base_model,
225         Flatten(),
226         Dense(128, activation='relu'),
227         Dense(1, activation='sigmoid')
228     ])
229
230     # Compile with low learning rate for transfer learning
231     model.compile(
232         optimizer=Adam(learning_rate=0.0001),
233         loss='binary_crossentropy',
234         metrics=['accuracy']
235     )
236
237     return model, base_model
238 
```

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



רשות عمוקה (CNN with Transfer Learning and Fine Tuning)

בגרסה זו, לאחר אימונו הראשוני עם רשות הבסיס הקפואה, בוצע **Fine Tuning** לחלק העליון של הרשות.

.Trainable layer 540 הוגדרו כ-.
שכבות תחתונות נשארו קפואות, ואילו השכבות מעל 540 הוגדרו כ-.
שימוש בקצב למידה נמוך במיוחד (0.0001) מאפשר התאמת מדויקת לדאטasset.

Code:

```

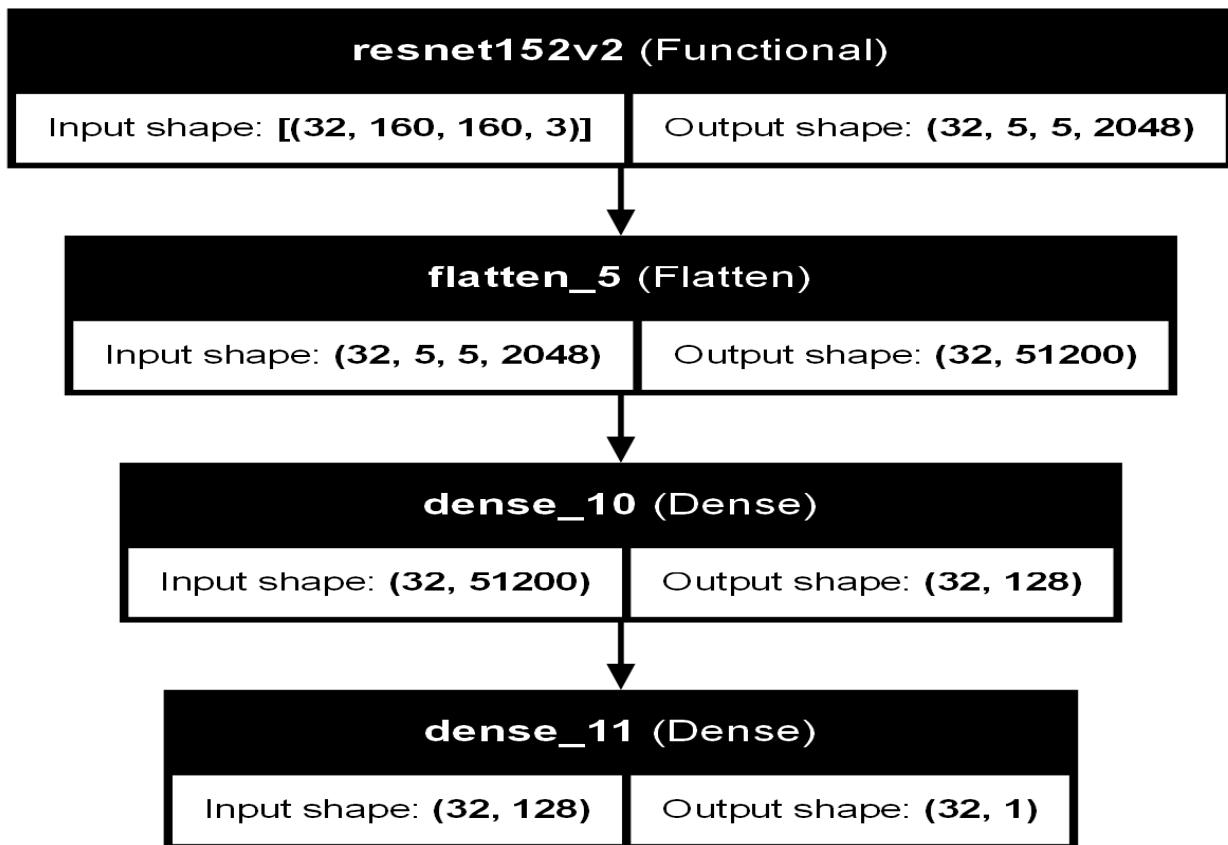
275 def create_cnn_with_transfer_learning_finetuned():
276     """
277         SOLUTION FOR QUESTION 1B (APPROACH 2): Transfer Learning with Fine-tuning
278         Report Section: Network 2B - CNN with Transfer Learning (Fine-tuned)
279         Creates CNN using pre-trained ResNet152V2 with fine-tuning for domain adaptation.
280         ARCHITECTURE DETAILS:
281         Base Network:
282             - Model: ResNet152V2 pre-trained on ImageNet
283             - Input Shape: 160x160x3 RGB images
284             - Include Top: False (remove original classification head)
285             - Fine-tuning Strategy: Unfreeze last layers (from layer 540 onwards)
286         Custom Classification Head:
287             - Flatten: Convert ResNet features to 1D vector
288             - Dense: 128 neurons, ReLU activation
289             - Output: 1 neuron, Sigmoid activation (binary classification)
290         FINE-TUNING STRATEGY:
291             - Phase 1: Train with frozen base (prevents catastrophic forgetting)
292             - Phase 2: Unfreeze top layers and retrain with learning rate
293             - Learning Rate: 0.0001
294         TRAINING CONFIGURATION:
295             - Optimizer: Adam with learning rate (0.0001)
296             - Loss Function: Binary Crossentropy
297             - Metrics: Accuracy
298         ADVANTAGES:
299             - Better adaptation to chest X-ray domain
300             - Higher potential accuracy than frozen approach
301             - Balances between overfitting and underfitting
302
303         REPORT USAGE: Compare performance with frozen and custom CNN approaches
304     """
312     # Load pre-trained ResNet152V2
313     inputs = tf.keras.Input(shape=(IMG_SIZE, 3), batch_size=BATCH_SIZE)
314     base_model = ResNet152V2(
315         weights='imagenet',
316         include_top=False,
317         input_tensor=inputs
318     )
319
320     # Initially freeze base model for stable training
321     base_model.trainable = False
322
323     # Create complete model with custom classification head
324     model = Sequential([
325         base_model,
326         Flatten(),
327         Dense(128, activation='relu'),
328         Dense(1, activation='sigmoid')
329     ])
330
331     # Compile with low learning rate for initial training
332     model.compile(
333         optimizer=Adam(learning_rate=0.0001),
334         loss='binary_crossentropy',
335         metrics=['accuracy']
336     )
337
338     # Enable fine-tuning: unfreeze top layers
339     base_model.trainable = True
340     fine_tune_at = 540 # Unfreeze from this layer onwards
341
342     # Keep early layers frozen to preserve generic features
343     for layer in base_model.layers[:fine_tune_at]:
344         layer.trainable = False
345
346     # Re-compile with learning rate for fine-tuning
347     model.compile(
348         optimizer=Adam(learning_rate=0.0001),
349         loss='binary_crossentropy',
350         metrics=['accuracy']
351     )
352
353     return model, base_model
354
  
```

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 הרשותות, ולצין בדוח את אלגוריתם האימון, מס' RATE LEARNING, גודל BATCH-MINI.

לגביה הרשות מבוססת TRANSFER LEARNING, יש להציג שני אופני אימון: א) פרמטרי רשות הבסיס מוקפים ורק השכבות שהוספו מתעדכנות באימון. ב) בו מעודכנים גם

חלק – לבחירתכם – של שכבות רשות הבסיס. איזה מהשיטות הביאה ל走出去ים טובים יותר מבחינת **ACCURACY**? המשיכו בסעיפים הבאים רק עם השיטה הטובה מבין השתיים.

2. עבור שתי הרשותות שתכוננתם צירו את גраф הביצועים, PRECISION - RECALL באשר כל נקודה בגרף תחושב עבור דמת סף שונה (ביחס להסתברות שפמיקה הרשות) להחלטה על דוגמא חיובית (כלומר עם דלקת ריאות). הסף יהיה בטוחו 0.1 עד 0.9 בקפיצות של 0.05. סמננו גם על הגרף את נקודות SCORE - F שיחושבו מכל זוג ערכיהם של PRECISION ו- RECALL .

3. עבור איזה סף התקבל ערך SCORE - F הגבוה ביותר?

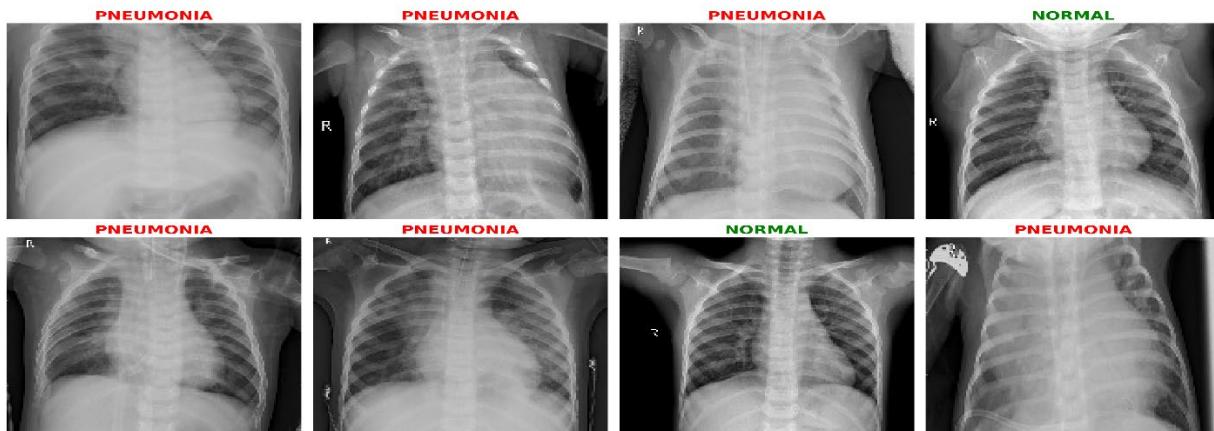
הסבירו: שימוש לב שההחלטה האם יש או אין בתמונה דלקת ריאות, מתקבלת ע"י הפעלת סף על ההסתברות שהרשות מוציאה (זכרו שהרשות מפיקה הסתברות רציפה בין 0 ל-1). מוצא הרשות מתקבל באמצעות הפונקציה predict(). במשימה 2 יש לשנות את הסף בתחום 0.1 עד 0.9 (בקפיצות של 0.05) ולהשך את ההחלטה בהתאם לסף: אם ההסתברות במושא הרשות גבוהה מהסף אז ההחלטה חיובית אחרת ההחלטה לדגמא: נניח שהסף הוא 0.3 והרשות מוציאה הסתברות 0.4, אז היוט ש: $0.3 < 0.4$ ההחלטה היא שיש דלקת ריאות. לכל סף מקבל ביצועים שונים של הרשות, אשר באמצעות חישוב צמד אחד של ערכי SCORE. כל נקודה בgraf RECALL-PRECISION תתקבל מערך שונה של הסף.

יבוא הנתונים והציגה ראשונית

הציגה ראשונית של הדאטאסט

בשלב זה יוצע ייבוא של הדאטאסט הכללי צילומי רנטגן של ריאות תקיןות ודלקת ריאות. הנתונים חולקו לשולשה סטים: אימון (Train), אימונות (Validation) ובדיקה (Test). לבחינה ראשונית של איקות הנתונים, הוציאו 8 תמונות אקראיות מהסט, דרך ניתן להבוחן בהבדלים בין התמונות התקינות לתמונות עם דלקת.

Dataset Sample Images - Chest X-Ray Pneumonia Detection



Dataset: NORMAL, PNEUMONIA • Image Size: 160x160 • Batch Size: 32

רשות 1 Transfer Learning ללא CNN – 1

Code:

```

143 def create_cnn_without_transfer_learning():
144     """
145         SOLUTION FOR QUESTION 1A: Custom CNN Architecture
146         Report Section: Network 1 - CNN without Transfer Learning
147
148         Creates a custom CNN model from scratch for binary pneumonia classification.
149
150         ARCHITECTURE DETAILS:
151             - Input Layer: 160x160x3 RGB chest X-ray images
152             - Convolutional Blocks: 4 identical blocks with:
153                 * Conv2D: 32 filters, 3x3 kernel, ReLU activation
154                 * MaxPooling2D: 2x2 pool size, stride=2
155             - Feature Extraction: Flatten layer converts 3D to 1D
156             - Classification Head:
157                 * Dense: 128 neurons, ReLU activation
158                 * Output: 1 neuron, Sigmoid activation (binary classification)
159
160         TRAINING CONFIGURATION:
161             - Optimizer: Adam (default learning rate)
162             - Loss Function: Binary Crossentropy
163             - Metrics: Accuracy
164
165         REPORT USAGE: Include architecture diagram and parameter count
166         """

```

```

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)),

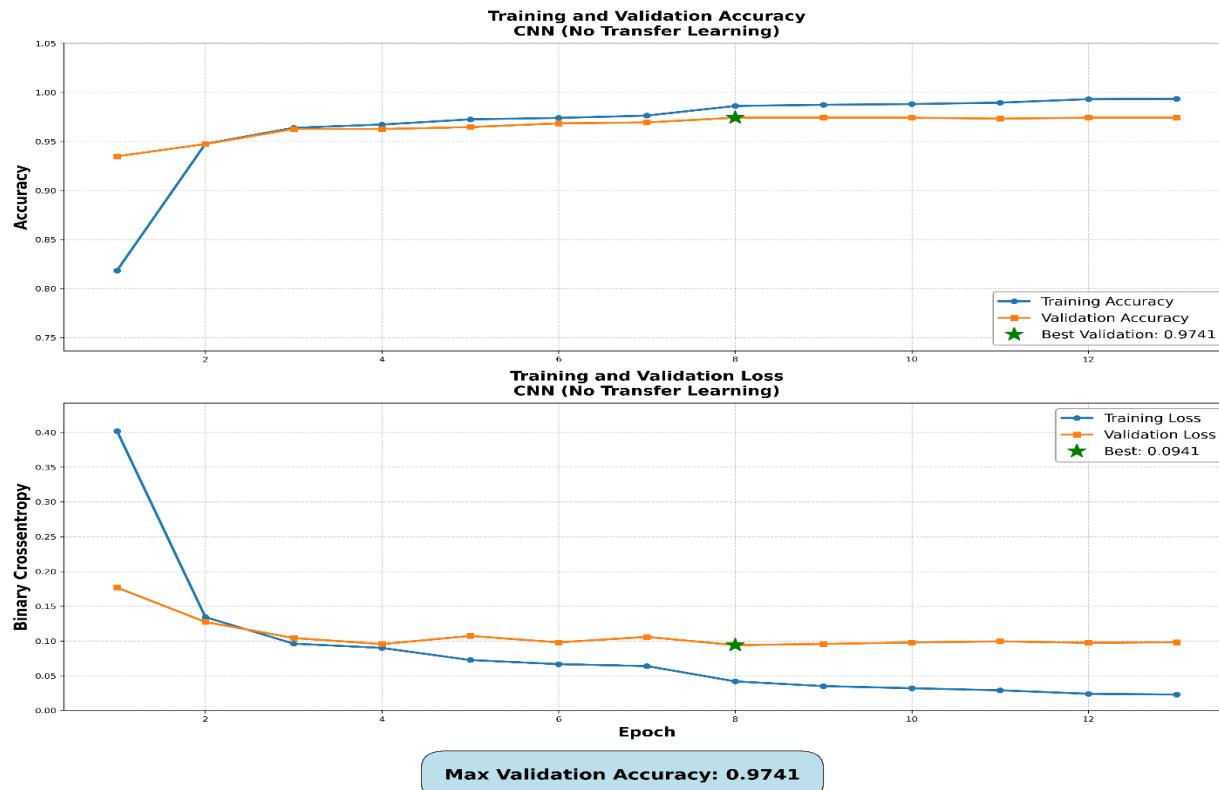
    # Feature Extraction
    Flatten(),

    # 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']
)

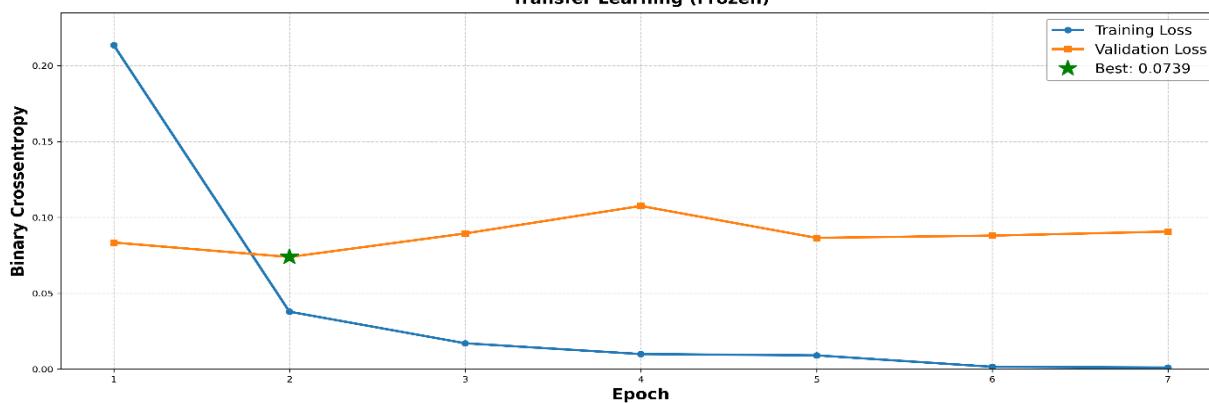
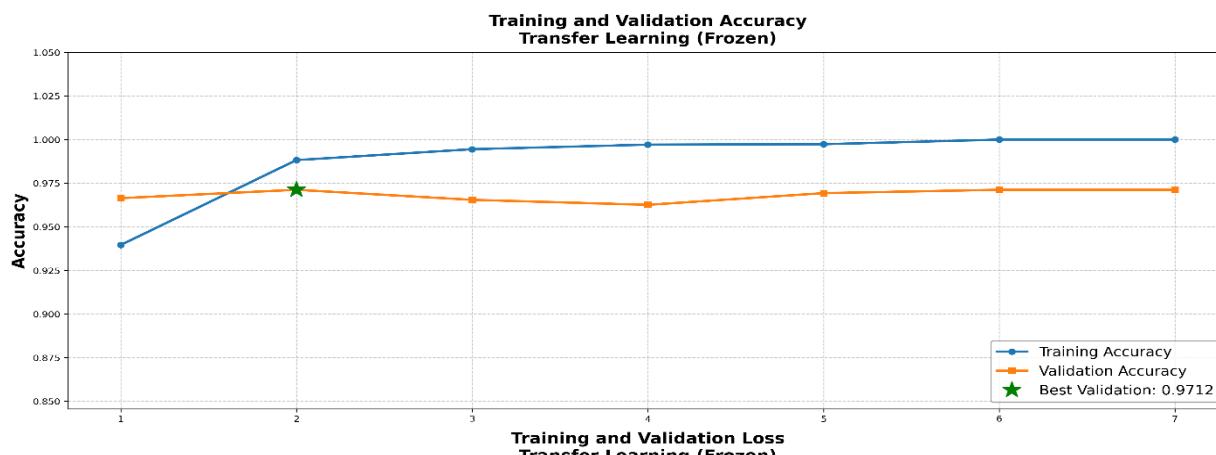
return model

```



רשות 2 Transfer Learning (Frozen) –

```
# -----  
# 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 (freeze 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))  
  
    # -----  
    # 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

רשות 3 Transfer Learning (Fine-Tuned) – 3

Code:

```

275 def create_cnn_with_transfer_learning_finetuned():
276     """
277         SOLUTION FOR QUESTION 1B (APPROACH 2): Transfer Learning with Fine-tuning
278         Report Section: Network 2B - CNN with Transfer Learning (Fine-tuned)
279         Creates CNN using pre-trained ResNet152V2 with fine-tuning for domain adaptation.
280         ARCHITECTURE DETAILS:
281             Base Network:
282                 - Model: ResNet152V2 pre-trained on ImageNet
283                 - Input Shape: 160x160x3 RGB images
284                 - Include Top: False (remove original classification head)
285                 - Fine-tuning Strategy: Unfreeze last layers (from layer 540 onwards)
286             Custom Classification Head:
287                 - Flatten: Convert ResNet features to 1D vector
288                 - Dense: 128 neurons, ReLU activation
289                 - Output: 1 neuron, Sigmoid activation (binary classification)
290         FINE-TUNING STRATEGY:
291                 - Phase 1: Train with frozen base (prevents catastrophic forgetting)
292                 - Phase 2: Unfreeze top layers and retrain with learning rate
293                 - Learning Rate: 0.0001
294         TRAINING CONFIGURATION:
295                 - Optimizer: Adam with learning rate (0.0001)
296                 - Loss Function: Binary Crossentropy
297                 - Metrics: Accuracy
298         ADVANTAGES:
299                 - Better adaptation to chest X-ray domain
300                 - Higher potential accuracy than frozen approach
301                 - Balances between overfitting and underfitting
302
303         REPORT USAGE: Compare performance with frozen and custom CNN approaches
304     """

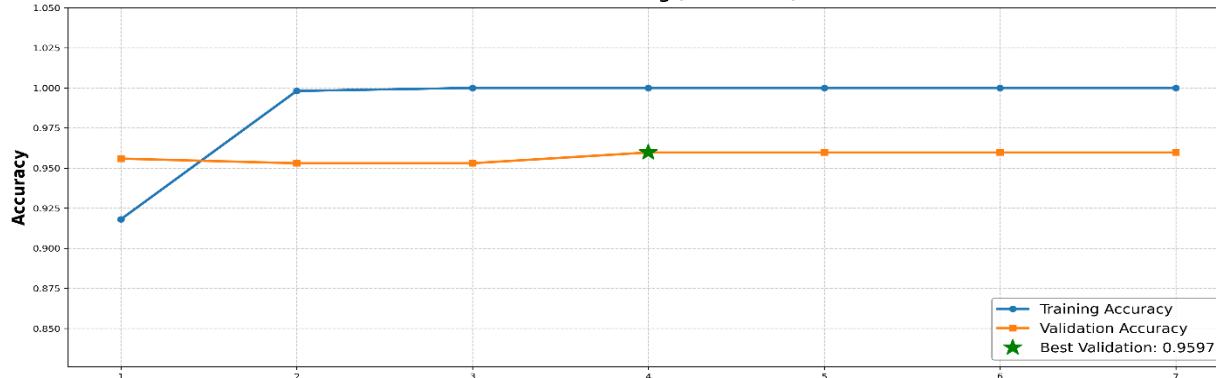
```

```

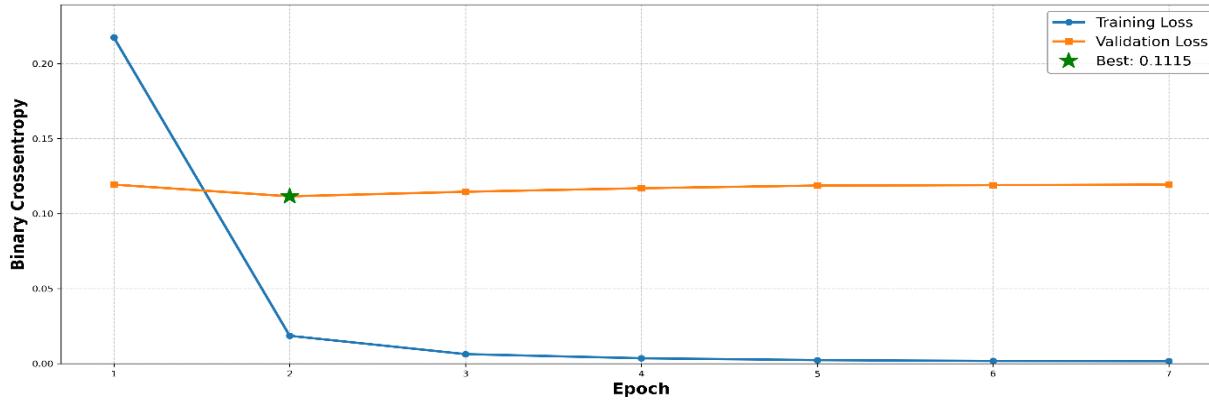
312     # Load pre-trained ResNet152V2
313     inputs = tf.keras.Input(shape=(IMG_SIZE, 3), batch_size=BATCH_SIZE)
314     base_model = ResNet152V2(
315         weights='imagenet',
316         include_top=False,
317         input_tensor=inputs
318     )
319
320     # Initially freeze base model for stable training
321     base_model.trainable = False
322
323     # Create complete model with custom classification head
324     model = Sequential([
325         base_model,
326         Flatten(),
327         Dense(128, activation='relu'),
328         Dense(1, activation='sigmoid')
329     ])
330
331     # Compile with low learning rate for initial training
332     model.compile(
333         optimizer=Adam(learning_rate=0.0001),
334         loss='binary_crossentropy',
335         metrics=['accuracy']
336     )
337
338     # Enable fine-tuning: unfreeze top layers
339     base_model.trainable = True
340     fine_tune_at = 540 # Unfreeze from this layer onwards
341
342     # Keep early layers frozen to preserve generic features
343     for layer in base_model.layers[:fine_tune_at]:
344         layer.trainable = False
345
346     # Re-compile with learning rate for fine-tuning
347     model.compile(
348         optimizer=Adam(learning_rate=0.0001),
349         loss='binary_crossentropy',
350         metrics=['accuracy']
351     )
352
353     return model, base_model
354

```

Training and Validation Accuracy
Transfer Learning (Fine-tuned)



Training and Validation Loss
Transfer Learning (Fine-tuned)



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) (Frozen) עם שכבות קופאות בלבד (Frozen)
- מודל עם Transfer Learning עם שכבות COPA בלבד (Frozen)
- מודל עם Transfer Learning עם שכבות מהשכבה ה-540 ומעלה (Fine Tuning)

1.2 - פרמטרי האימון (אחדים לכל המודלים):

פרמטר	ערך בריית מיחל / Frozen)	Fine-Tuned
מספר אפוקים (epochs)	20	20
גודל מיני-בצ' (mini-batch)	32	32
אופטימיזר	Adam	Adam
Learning Rate	0.0001	0.0001

1.3 - הסבר על שני אופני האימון ב:

. Training Frozen:

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

. Fine Tuning:

בגישה זו, בנוסף לשכבות הערליונות, נפתחו לעדכונו 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 מאפשר לרשת ללמידה יותר מבליל "לשכח" את הידע שנרכש באימון המקורי.

1.4 - ממצאים ראשוניים (Accuracy):

הגישה של – Transfer Learning בין אם עם שכבות קבועות ובין אם עם – Fine Tuning. מאפשרת למודלים למודד תכונות חזותיות עמוקות ומשמעויות מותוך התמונות, תוך ניצול ידע קודם מרשותו שאומנו על DATA (כמו ImageNet). רשות ה Fine-Tuned השינה את התוצאה הטובה ביותר, בזכות במונחי דיוק (Accuracy), האפשרות לעדכן חלק מהשכבות העמוקות בראש הבסיס ולכוון את הלמידה לדעת הספציפי של המשימה.

לעומת זאת, המודל שנבנה ללא Transfer Learning נתן ביצועים טובים יחסית באימון, אך קיים חשש לכך – Overfitting – כלומר התאמת יתרה לנוטני האימון והקללה חלשה על DATA חדש.

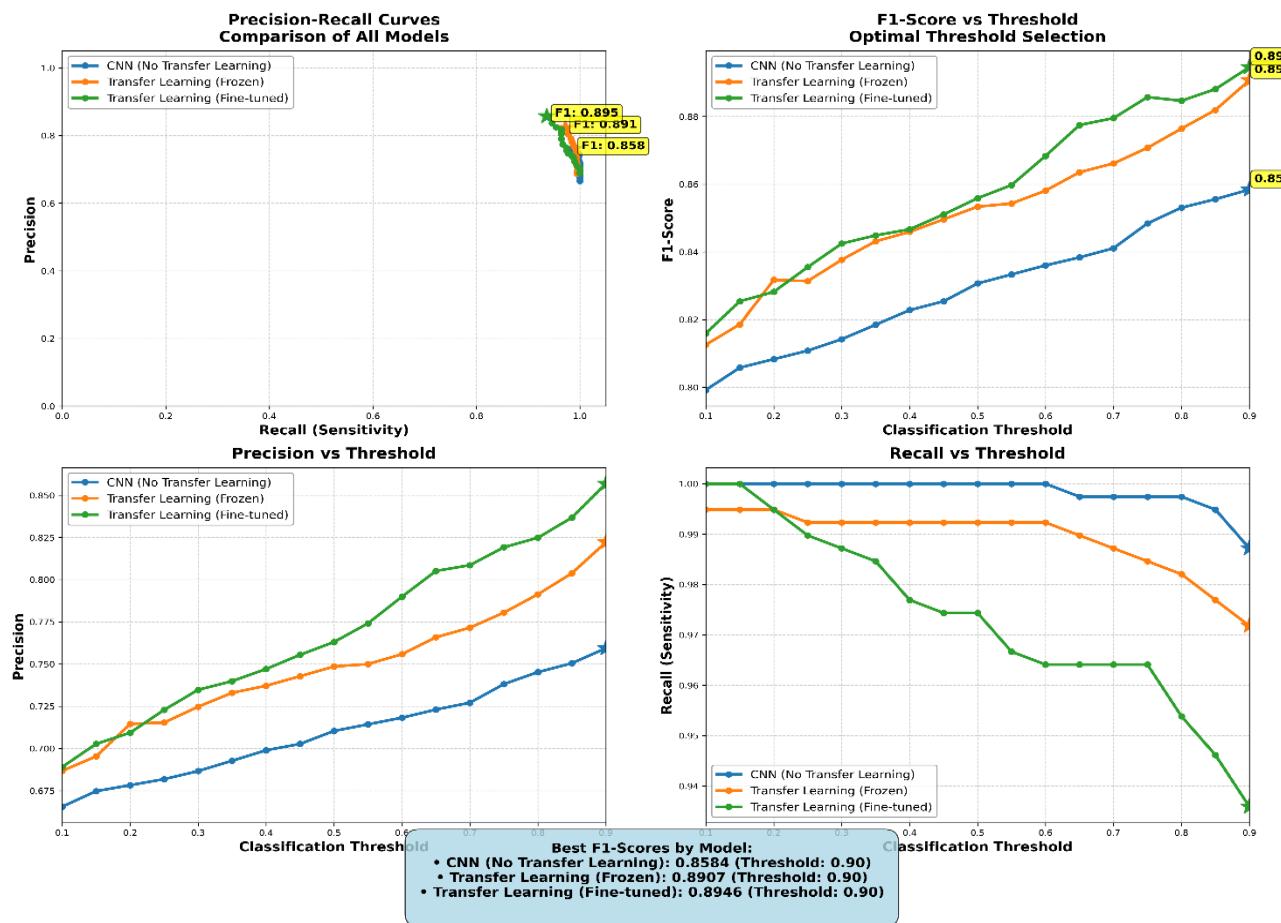
לאור זאת, נבחר המשיך לשלבים הבאים (סעיפים 2–3) עם המודל המבוסס Accuracy. שהראה יתרון ברור מבחן.

```

241 def evaluate_with_thresholds(model, test_ds, model_name):
242     """
243         QUESTION 2C - Precision-Recall Analysis with Multiple Thresholds
244
245         Performs comprehensive threshold analysis to find optimal classification thresholds.
246         This analysis is crucial for medical diagnosis applications where precision and recall
247         balance is critical for clinical decision making.
248
249     Threshold Analysis:
250     - Tests thresholds from 0.1 to 0.9 with steps of 0.05
251     - Calculates precision, recall, and F1-score for each threshold
252     - Identifies optimal threshold based on F1-score maximization
253     - Provides detailed performance metrics
254
255     Clinical Relevance:
256     - Higher thresholds: Better precision (fewer false positives)
257     - Lower thresholds: Better recall (fewer false negatives)
258     - Optimal threshold: Best balance for pneumonia detection
259
260     Args:
261         model: Trained CNN model
262         test_ds: Test dataset
263         model_name: Model name for logging
264
265     Returns:
266         Dictionary containing:
267             - thresholds: Array of tested thresholds
268             - precisions: Precision values for each threshold
269             - recalls: Recall values for each threshold
270             - f1_scores: F1-scores for each threshold
271             - best_threshold: Optimal threshold value
272             - best_f1: Maximum F1-score achieved
273
274     print(f"\n🔍 Evaluating {model_name} with different thresholds...")
275
276     # Get predictions and true labels
277     y_true = []
278     y_pred_prob = []
279
280     for images, labels in test_ds:
281         y_true.extend(labels.numpy())
282         predictions = model.predict(images, verbose=0)
283         y_pred_prob.extend(predictions.flatten())
284
285     y_true = np.array(y_true)
286     y_pred_prob = np.array(y_pred_prob)
287
288     # Calculate precision and recall for different thresholds
289     thresholds = np.arange(0.1, 0.95, 0.05) # From 0.1 to 0.9 with steps of 0.05
290     precisions = []
291     recalls = []
292     f1_scores = []
293
294     print(f"💡 Testing {len(thresholds)} thresholds: {thresholds}")
295
296     for threshold in thresholds:
297         # Apply threshold
298         y_pred_binary = (y_pred_prob >= threshold).astype(int)
299
300         # Calculate metrics
301         precision = precision_score(y_true, y_pred_binary, zero_division=0)
302         recall = recall_score(y_true, y_pred_binary, zero_division=0)
303         f1 = f1_score(y_true, y_pred_binary, zero_division=0)
304
305         precisions.append(precision)
306         recalls.append(recall)
307         f1_scores.append(f1)
308
309     # Find best threshold based on F1-score
310     best_f1_idx = np.argmax(f1_scores)
311     best_threshold = thresholds[best_f1_idx]
312     best_f1 = f1_scores[best_f1_idx]
313     best_precision = precisions[best_f1_idx]
314     best_recall = recalls[best_f1_idx]
315
316     print(f"\n💡 Best Results for {model_name}:")
317     print(f"👉 Best Threshold: {best_threshold:.2f}")
318     print(f"👉 Best F1-Score: {best_f1:.4f}")
319     print(f"👉 Precision at Best F1: {best_precision:.4f}")
320     print(f"👉 Recall at Best F1: {best_recall:.4f}")
321
322     return {
323         'thresholds': thresholds,
324         'precisions': precisions,
325         'recalls': recalls,
326         'f1_scores': f1_scores,
327         'best_threshold': best_threshold,
328         'best_f1': best_f1,
329         'best_precision': best_precision,
330         'best_recall': best_recall
331     }

```

ניתוח לפי סעיף 2:



מטרה:

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

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

דוגמה להסביר המשמעות של הסף:

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

לעומת זאת, אם הסף היה 0.5, אותה דוגמה תסוווג כבריאה (שלילי).
מכאן עולה כי העלאת הסף מעלה את **Precision** אך מורידה את **Recall** – ולהיפך.

תוצאות ניתוח F1-Score :

מודל	F1-Score מקסימלי	אופטימלי Threshold
Transfer Learning (Fine-Tuned)	0.8946	0.90
Transfer Learning (Frozen)	0.8907	0.90
No Transfer Learning	0.8584	0.90

הגרפים מראים שלושת המודלים הגיעו לביצוע השיא שלהם ב- $\text{Threshold} = 0.9$ אך F1 הציג את ערך Fine Tuning הגבוה ביותר.

מסקנה:

מודל **Transfer Learning** עם **Fine Tuning** הצליח לאזן ב�ורה הטובה ביותר בין דיוק (Precision) לריגישות – (Recall) גם כאשר סף החלטה גבוהה יחסית. זה יתרון משמעותי בהקשרים רפואיים, בהם **זהות מוקדם** (Recall) גבוהה (חשוב במיוחד בכך שאין צורך במניעת **חיבוביים שגויים** (Precision גבוהה)).

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 גבוהה, ריגישות מאוזנת, וGamification בהתאם לסטטוס קליניים משתנים.

משימה 3 – בדיקת השפעת האופטימיזציה והפעלת Early Stopping

משימה 3

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

- א. אלגוריתם SGD
- ב. אלגוריתם SGD עם MOMENTUM.
- ג. אלגוריתם ADAM
- ד. אלגוריתם RMSPROP

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 ביצועים באמצעות מנגנון זה?

במשימה זאת יש להציג את גרפּ ההתקבשות של תהליכי האימון לכל אחד מהסעיפים כולל TRAIN LOSS ו- VALIDATION LOSS, וכן TRAIN ACCURACY ו- VALIDATION ACCURACY. את באשר נעשה שימוש בסרט ולידציה שבכלל לפחות 50 תמונות עם דלקת ריאות (25 מכל סוג) ו- 50 תמונות ללא דלקת ריאות. סט הולידיצה ילקח מהתוך סט האימון (בר שיהוו קבוצות דומות של תמונות).

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

- מס' Epochs
- ערכי Learning Rate
- סוג ה- Optimizer

לצורך כך, הרצינו את הקוד על רשת CNN (ללא Transfer Learning) ועל רשת Optimizers 4 (Fine-Tuned) Learning SGD .1

Momentum עם SGD .2

Adam .3

RMSprop .4

השתמשנו בערכי 0.01, 0.001, 0.0001

. Epochs: 5, 10, 15 , 17

ונצרו גרפים עבור כל אופטימיזר, כאשר כל גרפּ מציג:

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

1) השפעת Epochs ו- Optimizer, Learning Rate

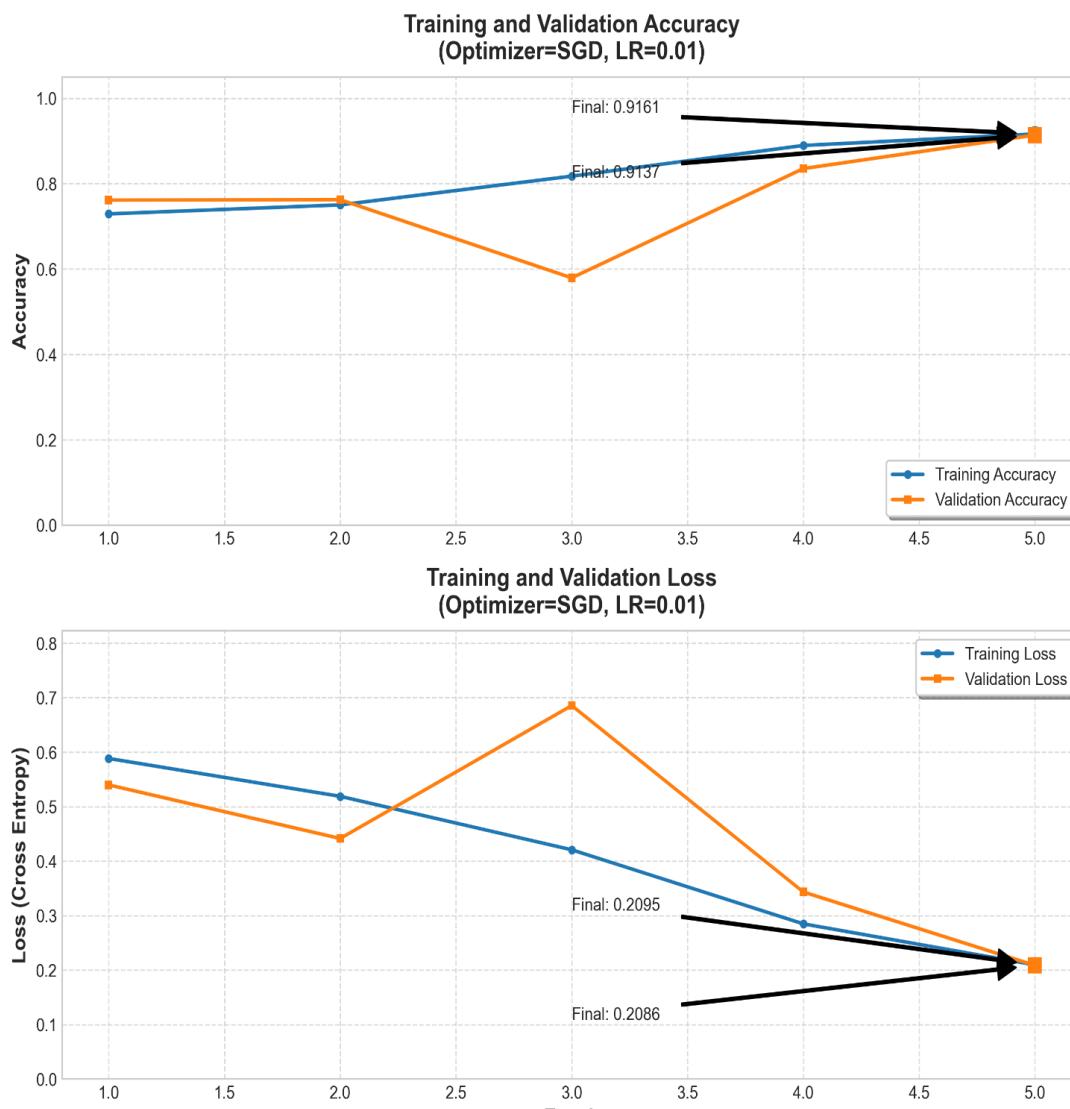
רשות CNN ללא Transfer learning

a. אלגוריתם SGD

בדקנו את השפעת SGD הקלasicי ללא Momentum.

➤ SGD - Graph for learning rate 0.01:

❖ Epochs 5:

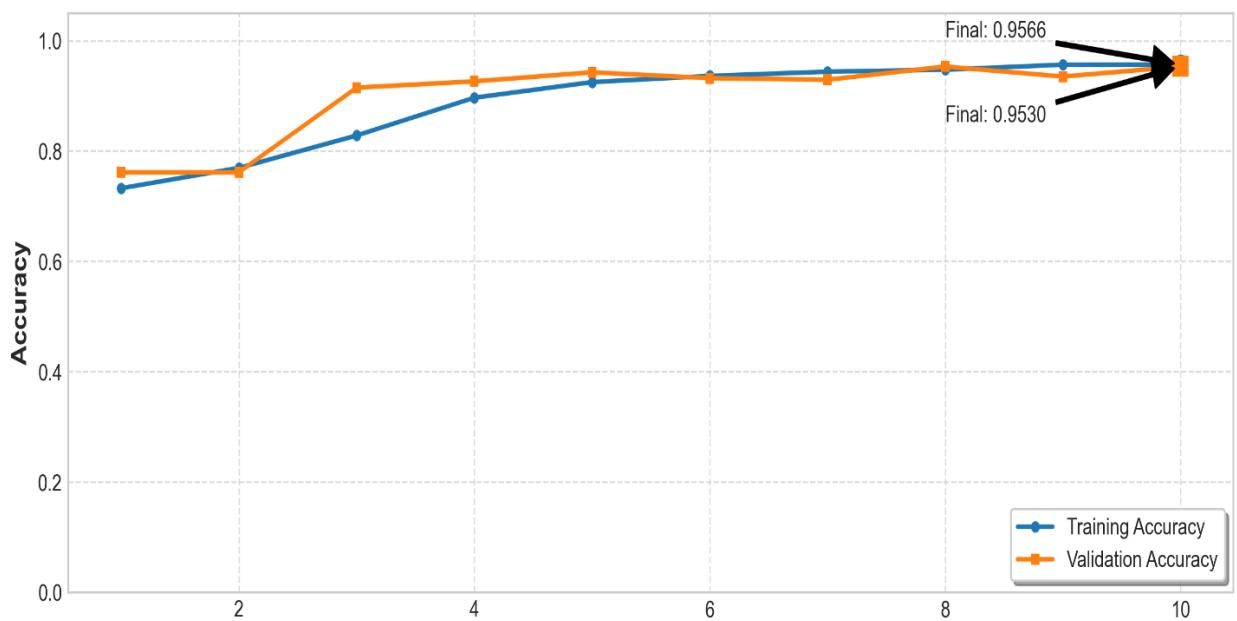


Summary Metrics:

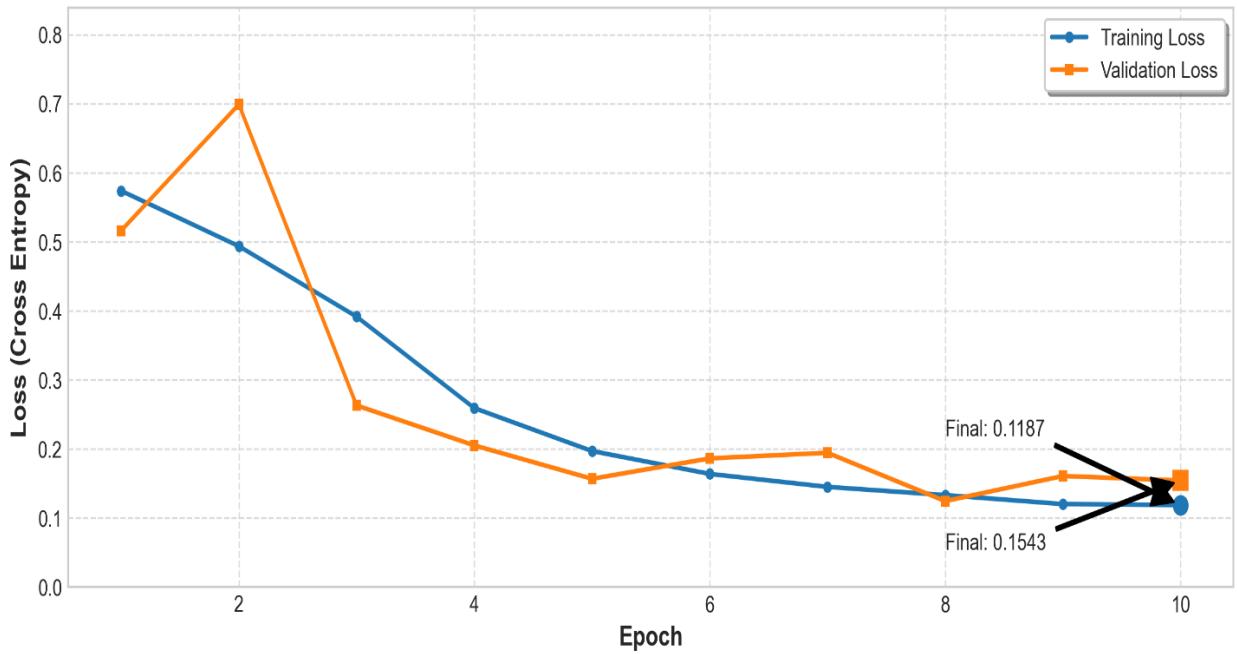
- Final Training Accuracy: 0.9161, Validation Accuracy: 0.9137
- Best Validation Accuracy: 0.9137 (Epoch 5)
- Total Images: 1043 training, 1043 validation

❖ Epochs 10:

Training and Validation Accuracy
(Optimizer=SGD, LR=0.01)



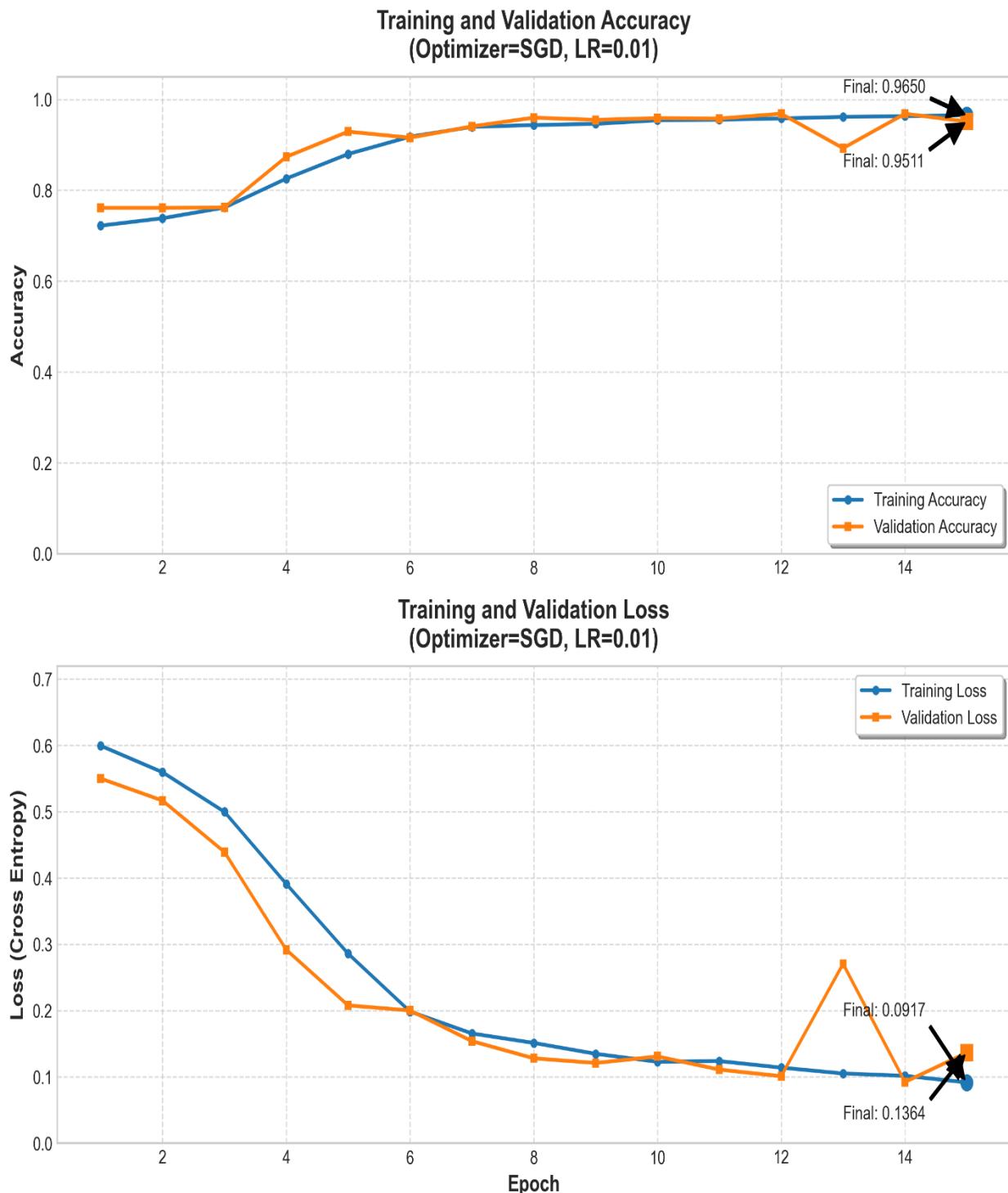
Training and Validation Loss
(Optimizer=SGD, LR=0.01)



Summary Metrics:

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

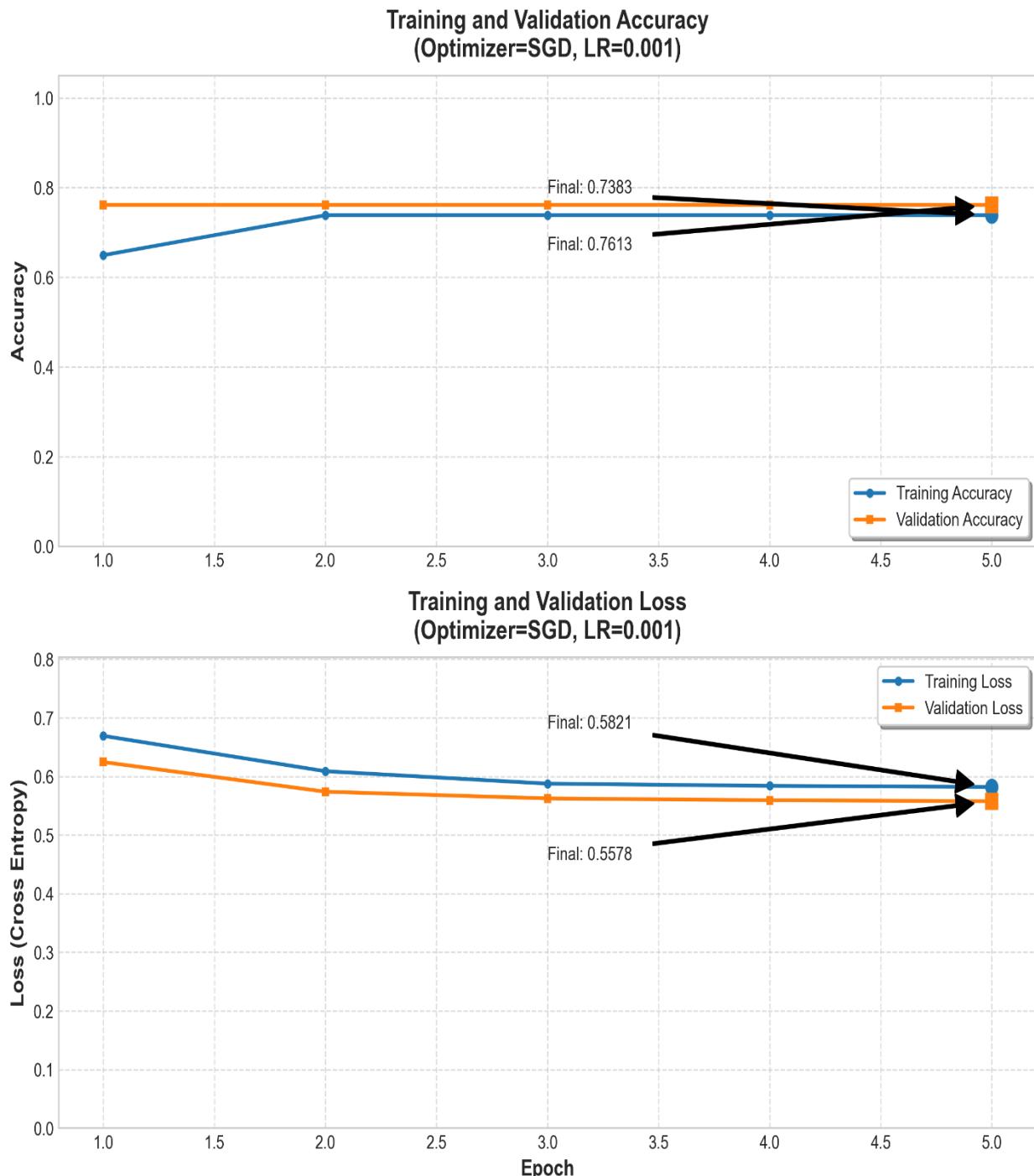
❖ Epochs 15:



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:

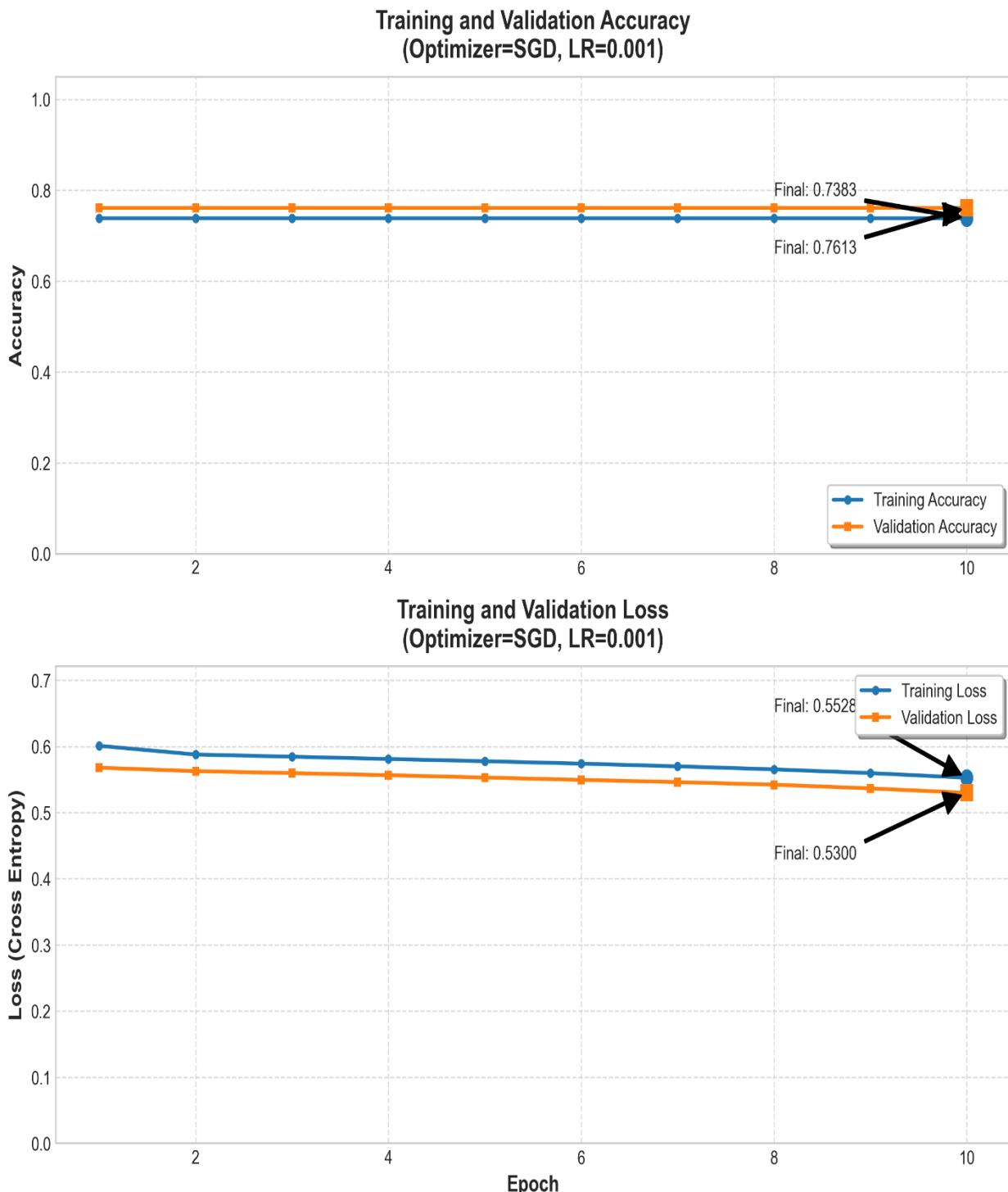
❖ Epochs 5:



Summary Metrics:

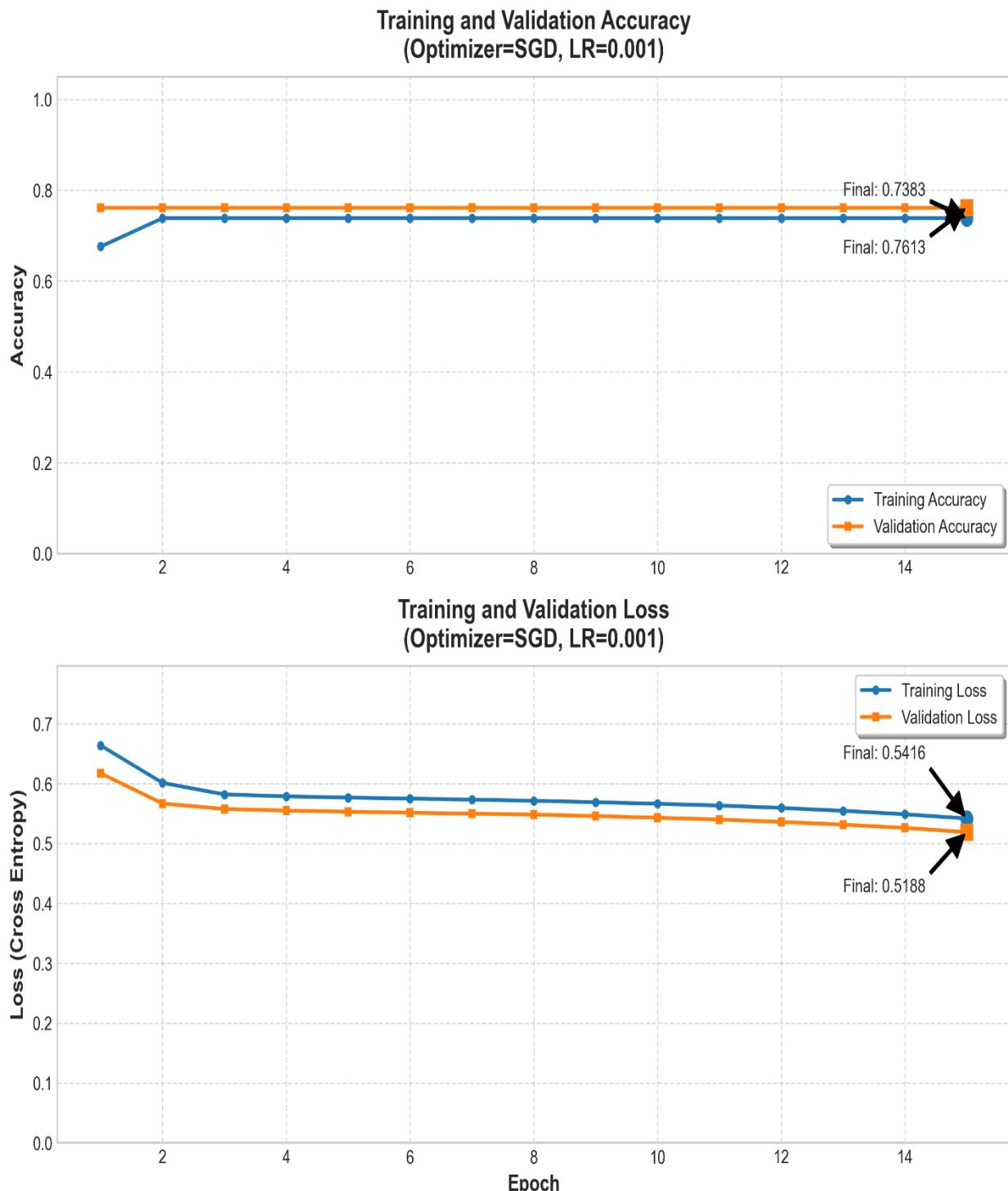
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

❖ 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

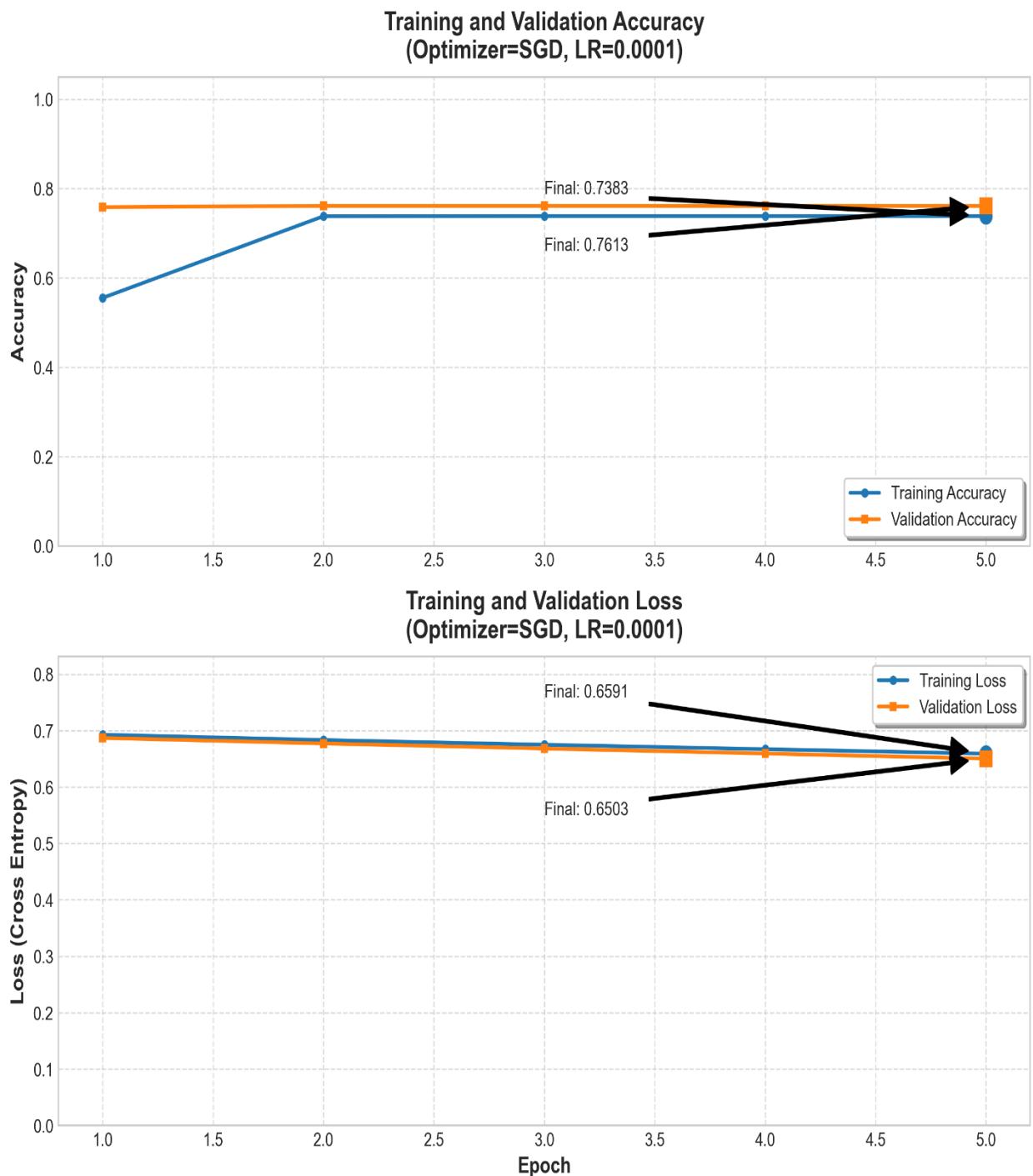
❖ Epochs 15:



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

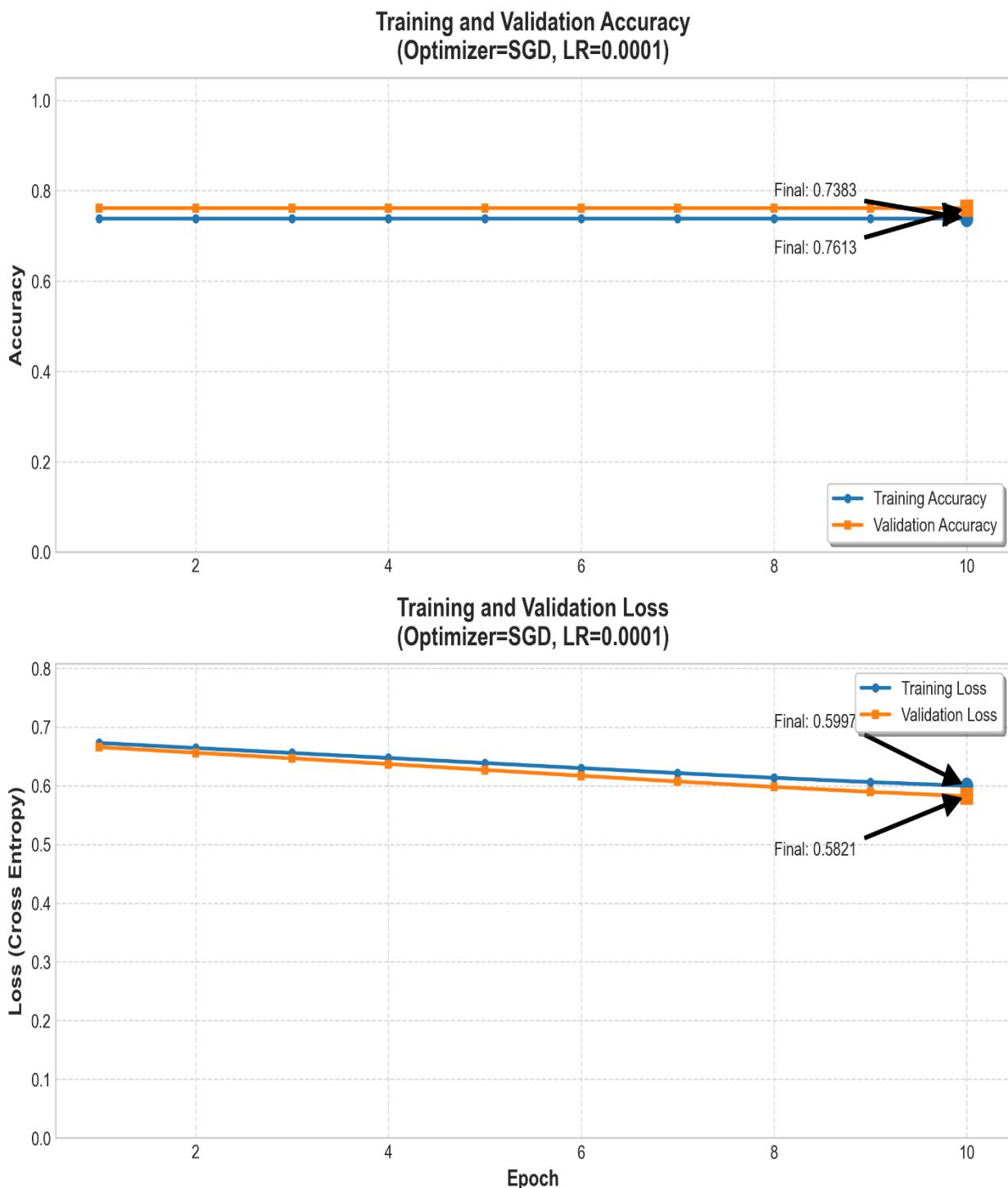
➤ SGD - Graph for learning rate 0.0001:

❖ Epochs 5:



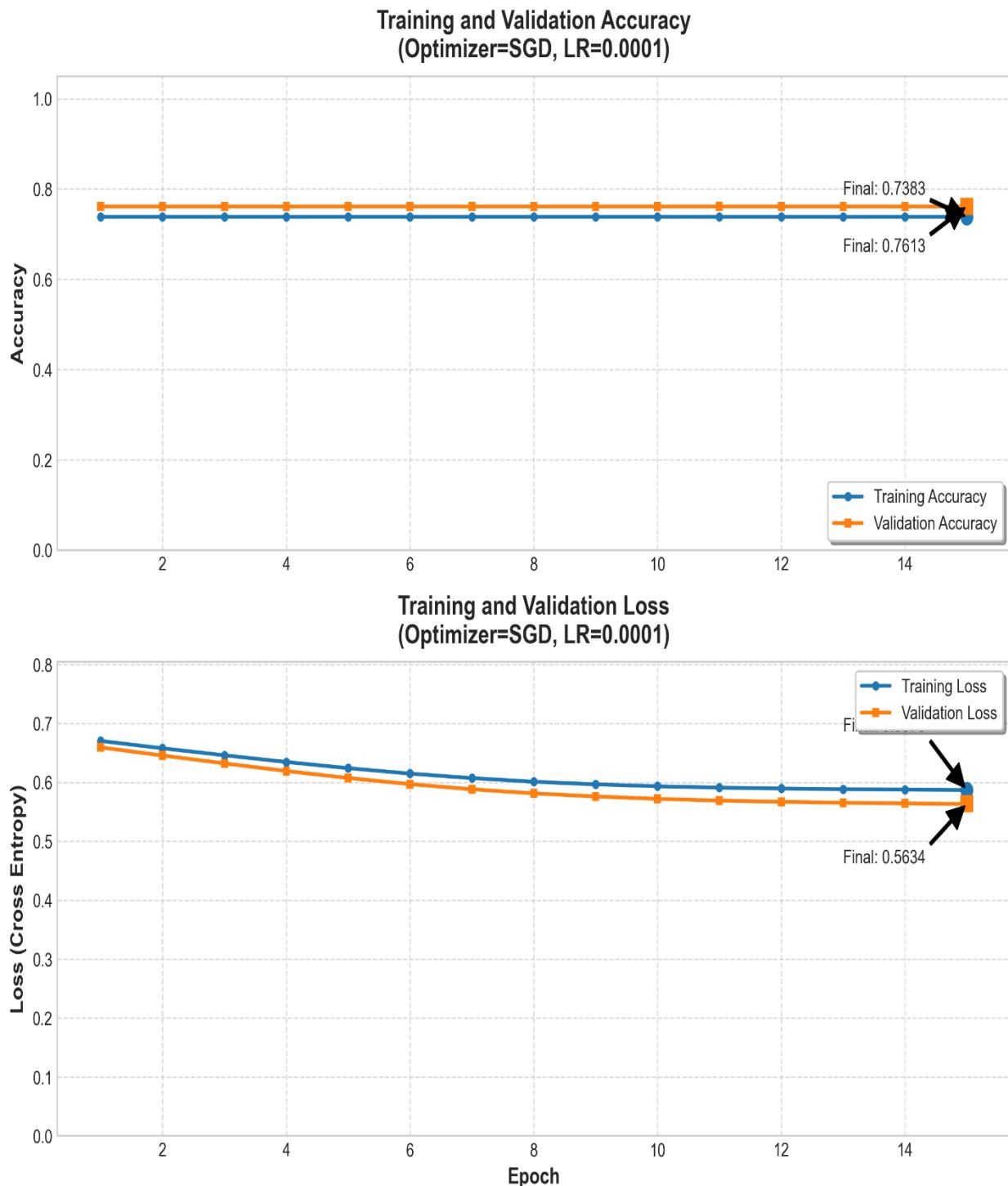
Summary Metrics:
 • Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
 • Best Validation Accuracy: 0.7613 (Epoch 2)
 • Total Images: 1043 training, 1043 validation

❖ 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:

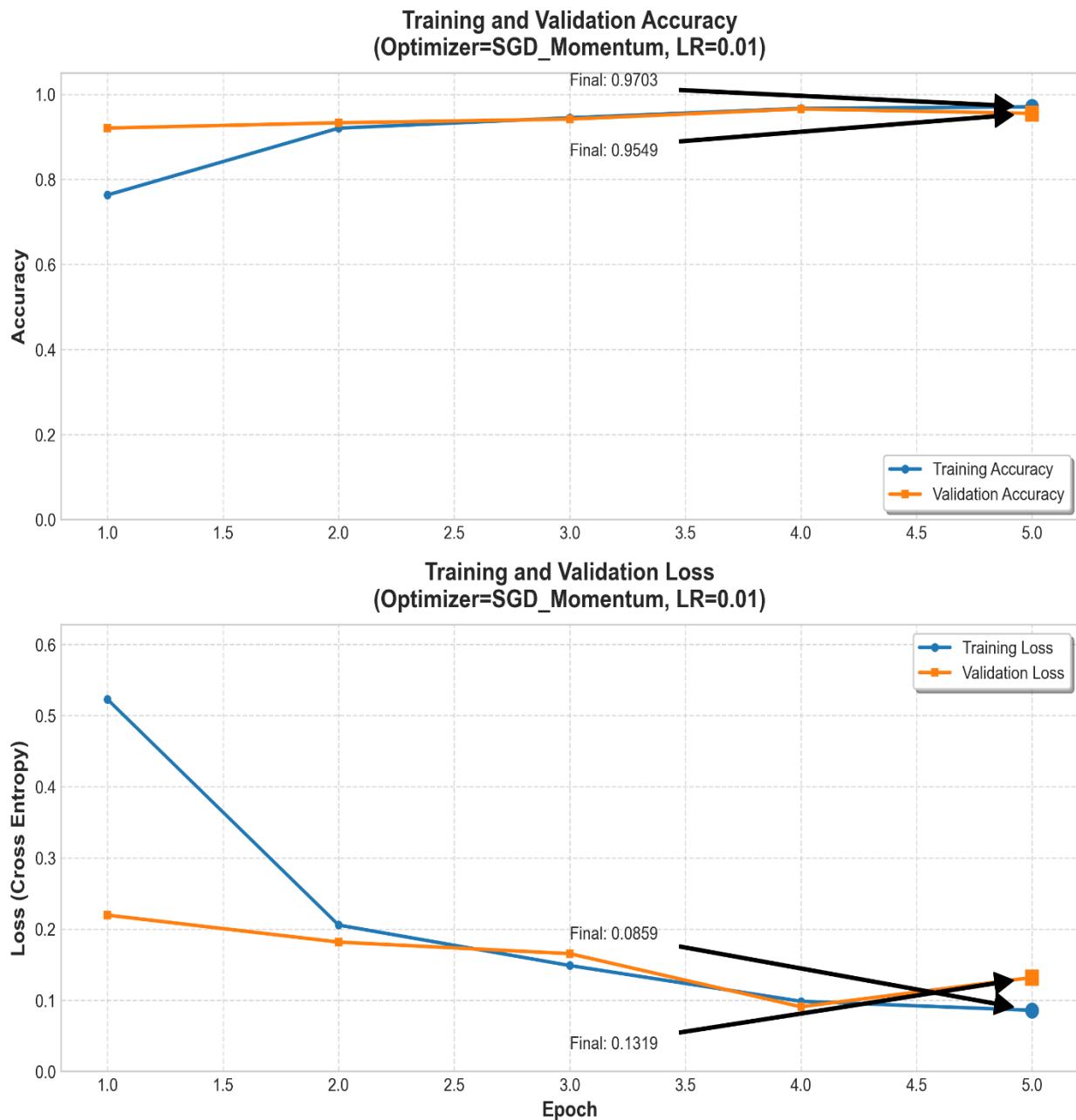
- Final Training Accuracy: 0.7383, Validation Accuracy: 0.7613
- Best Validation Accuracy: 0.7613 (Epoch 1)
- Total Images: 1043 training, 1043 validation

ב. אלגוריתם SGD עם Momentum

Momentum = 0.9

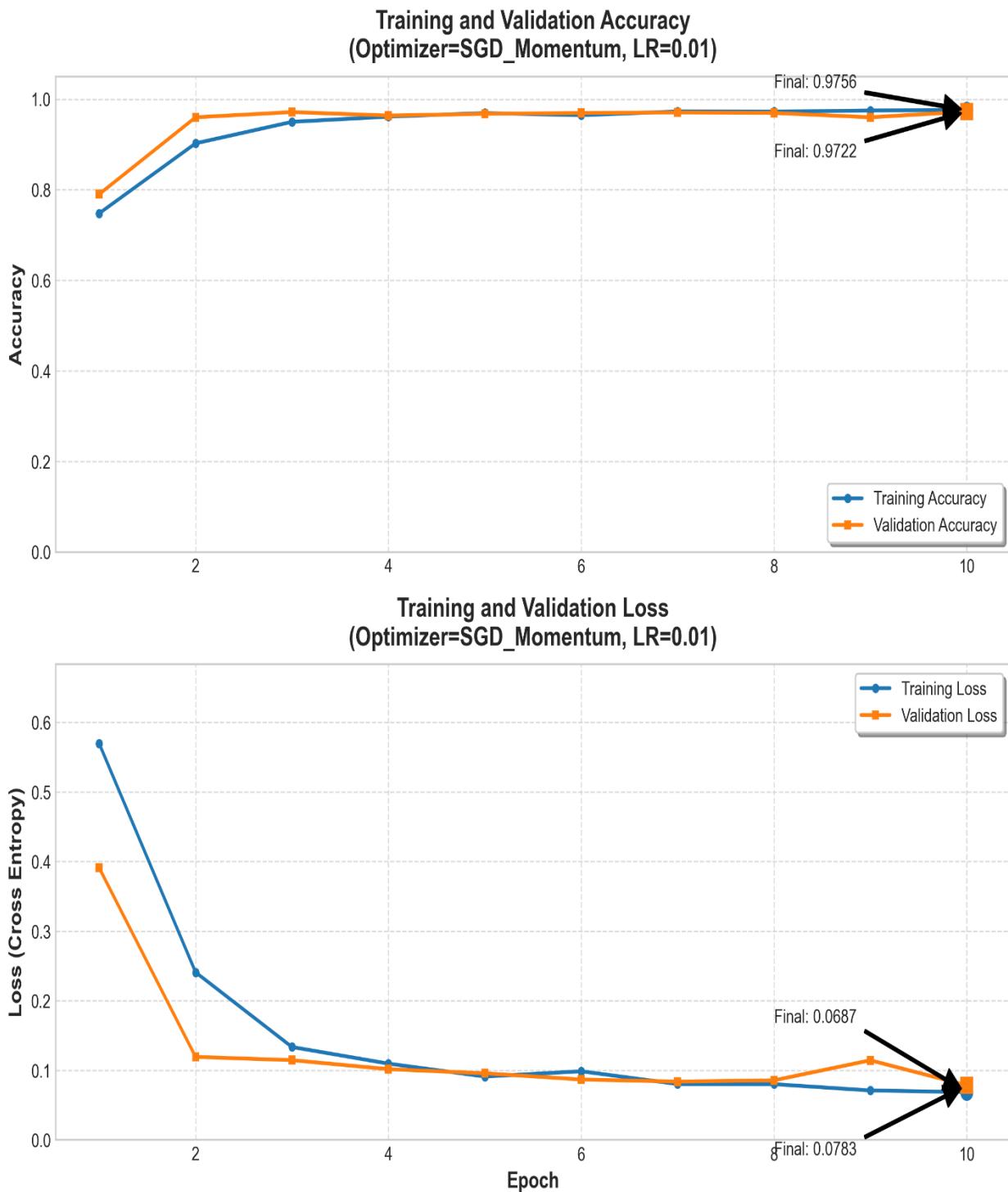
➤ SGD with Momentum - Graph for learning rate 0.01:

❖ Epochs 5:



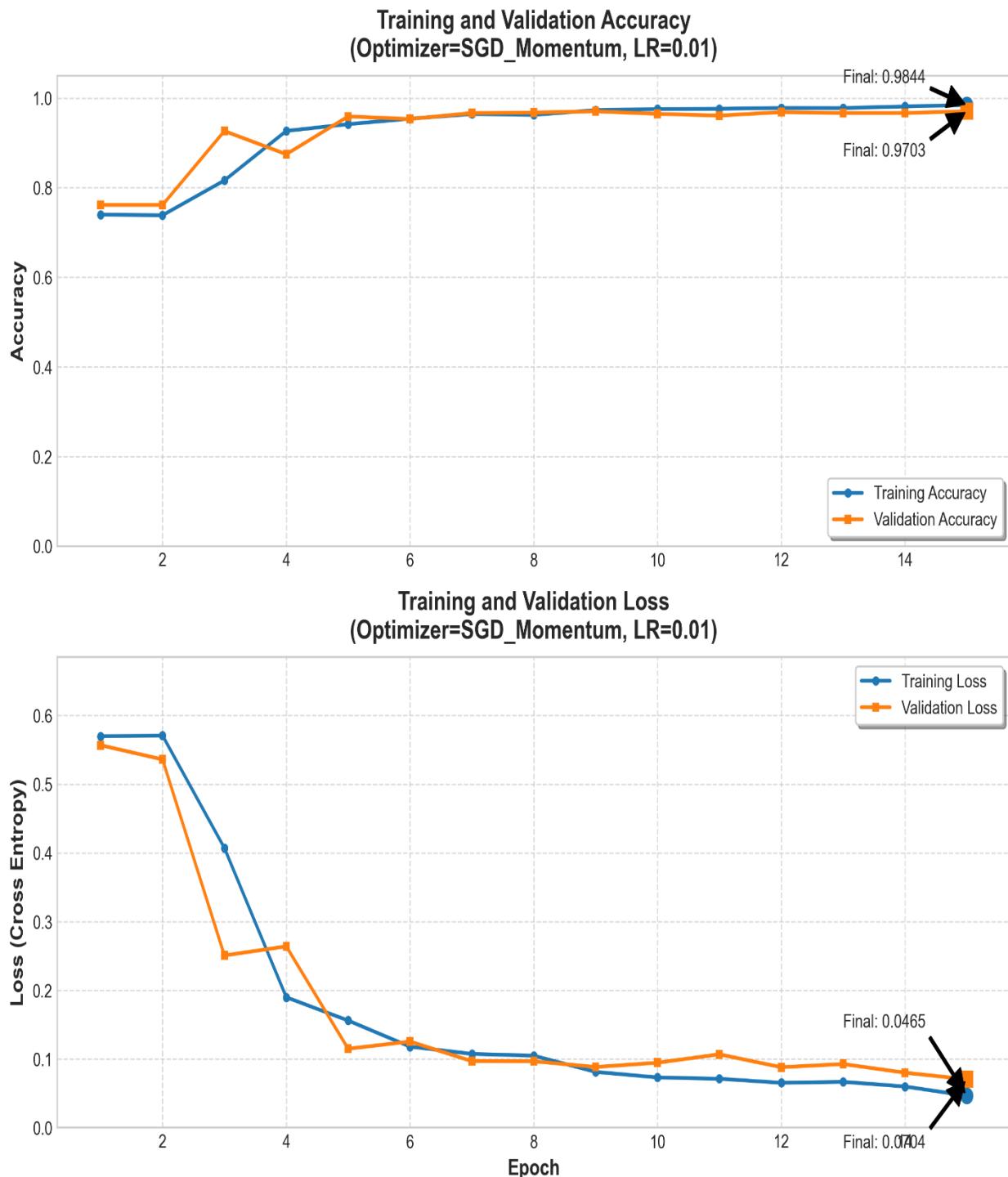
Summary Metrics:
 • Final Training Accuracy: 0.9703, Validation Accuracy: 0.9549
 • Best Validation Accuracy: 0.9655 (Epoch 4)
 • Total Images: 1043 training, 1043 validation

❖ 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

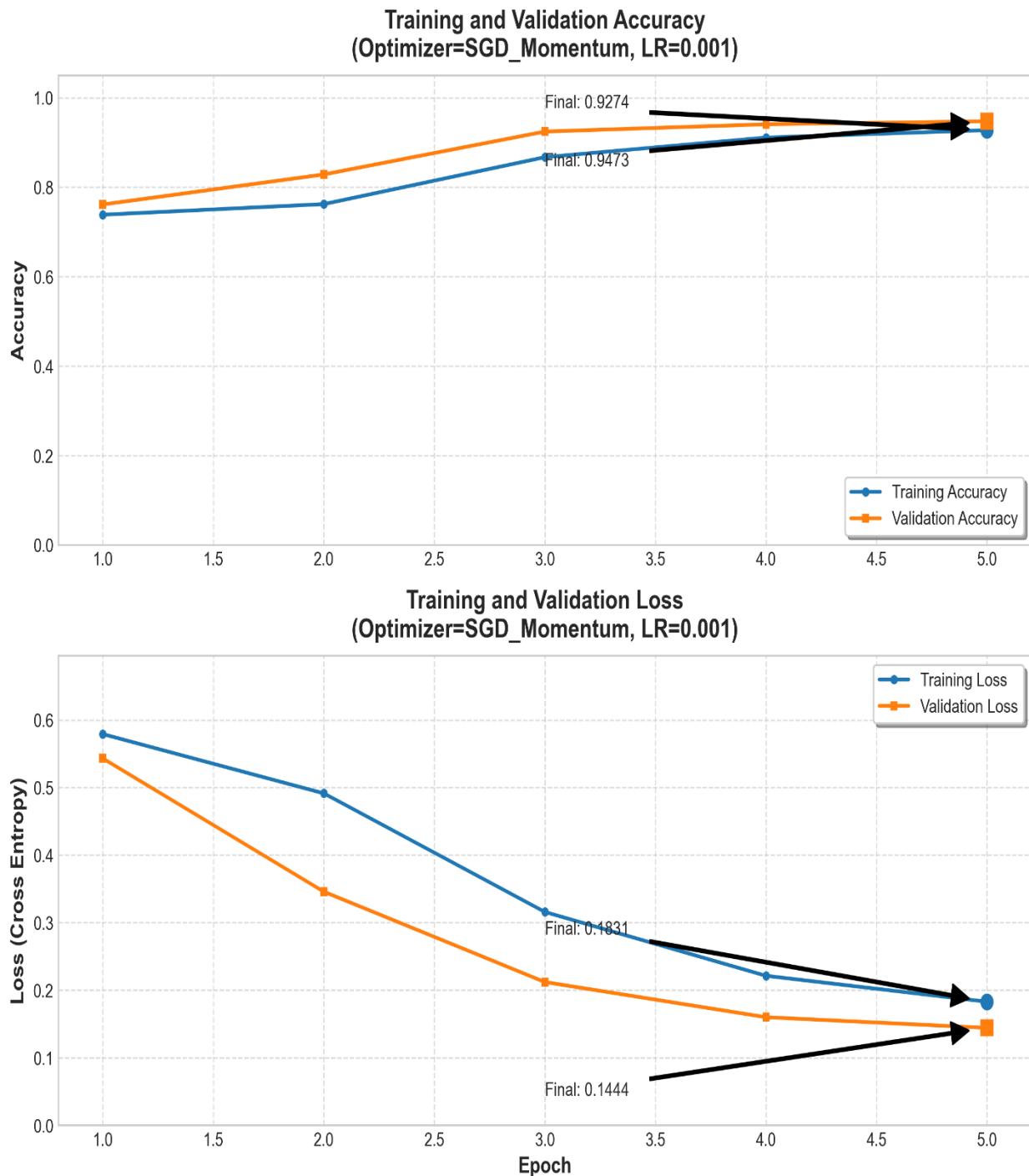
❖ Epochs 15:



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

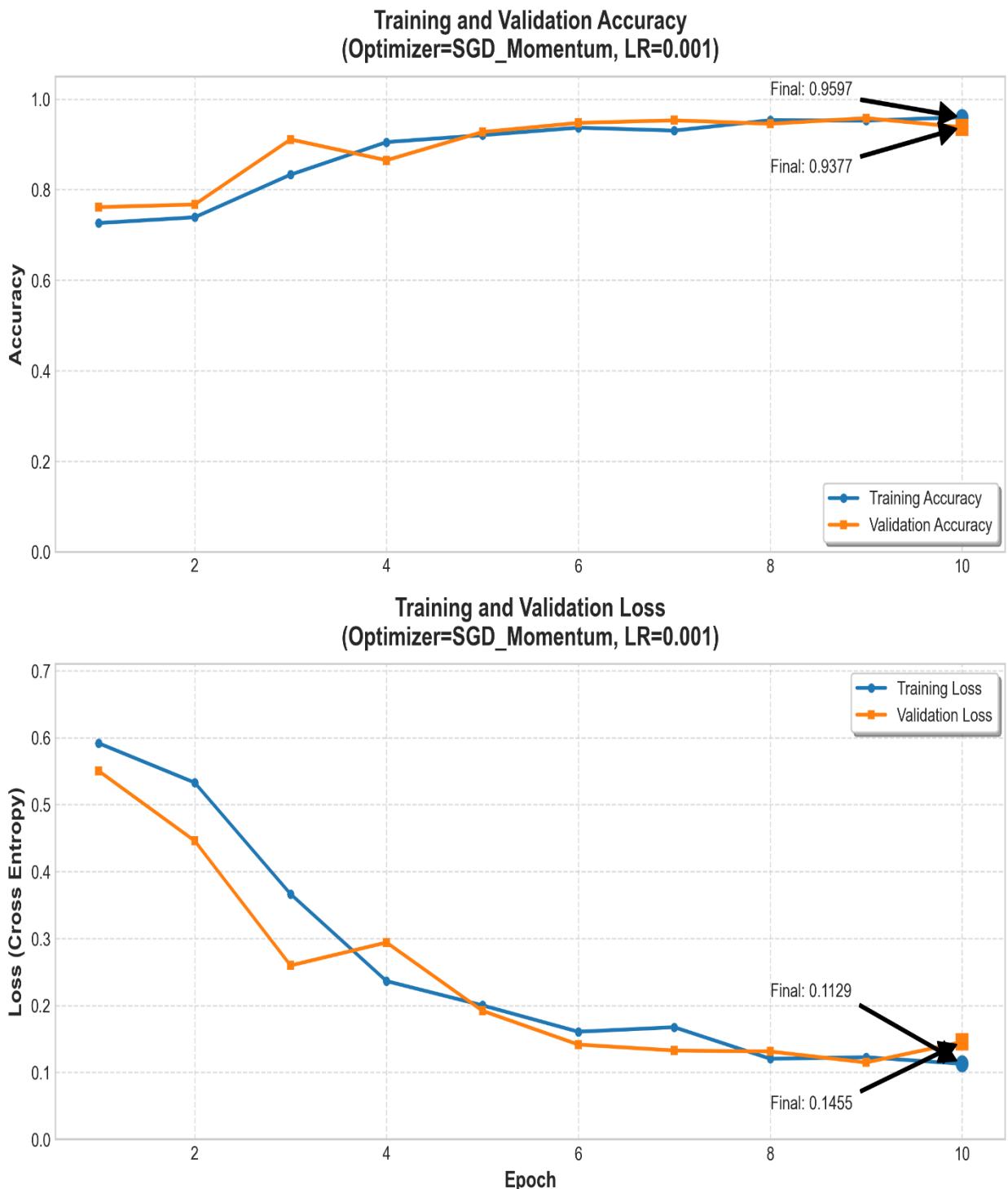
➤ SGD with Momentum - Graph for learning rate 0.001:

❖ Epochs 5:



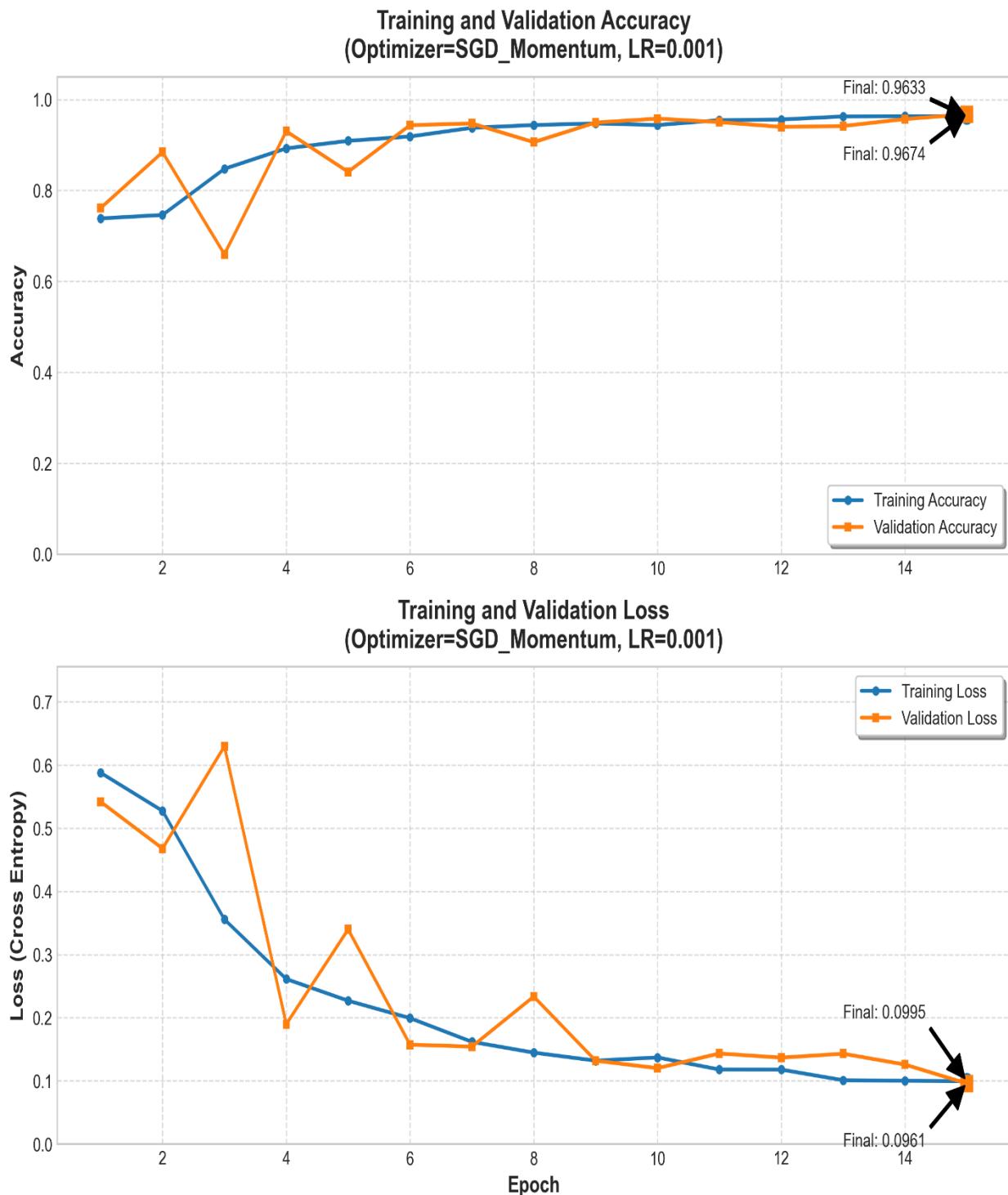
Summary Metrics:
 • Final Training Accuracy: 0.9274, Validation Accuracy: 0.9473
 • Best Validation Accuracy: 0.9473 (Epoch 5)
 • Total Images: 1043 training, 1043 validation

❖ 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

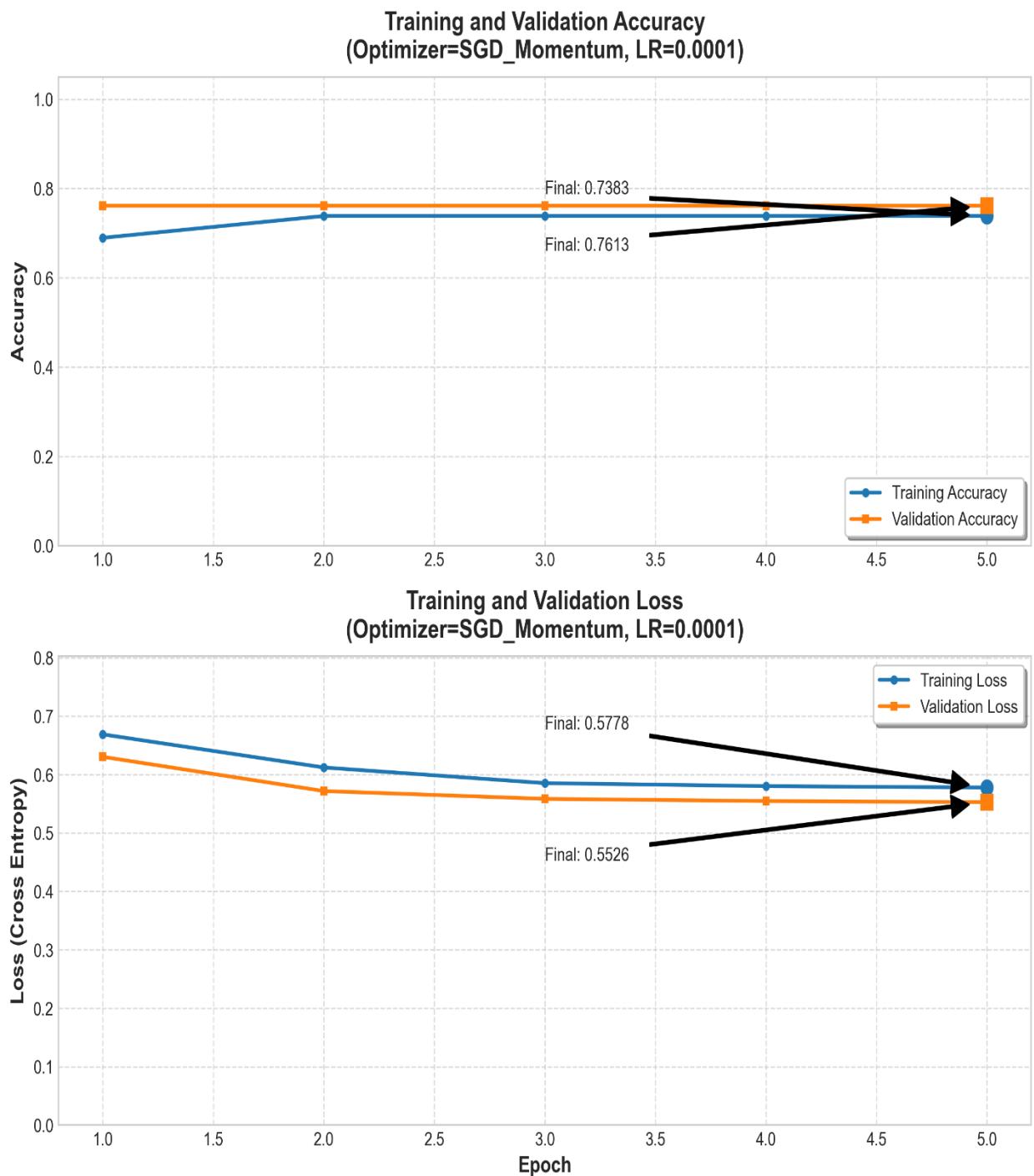
❖ Epochs 15:



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

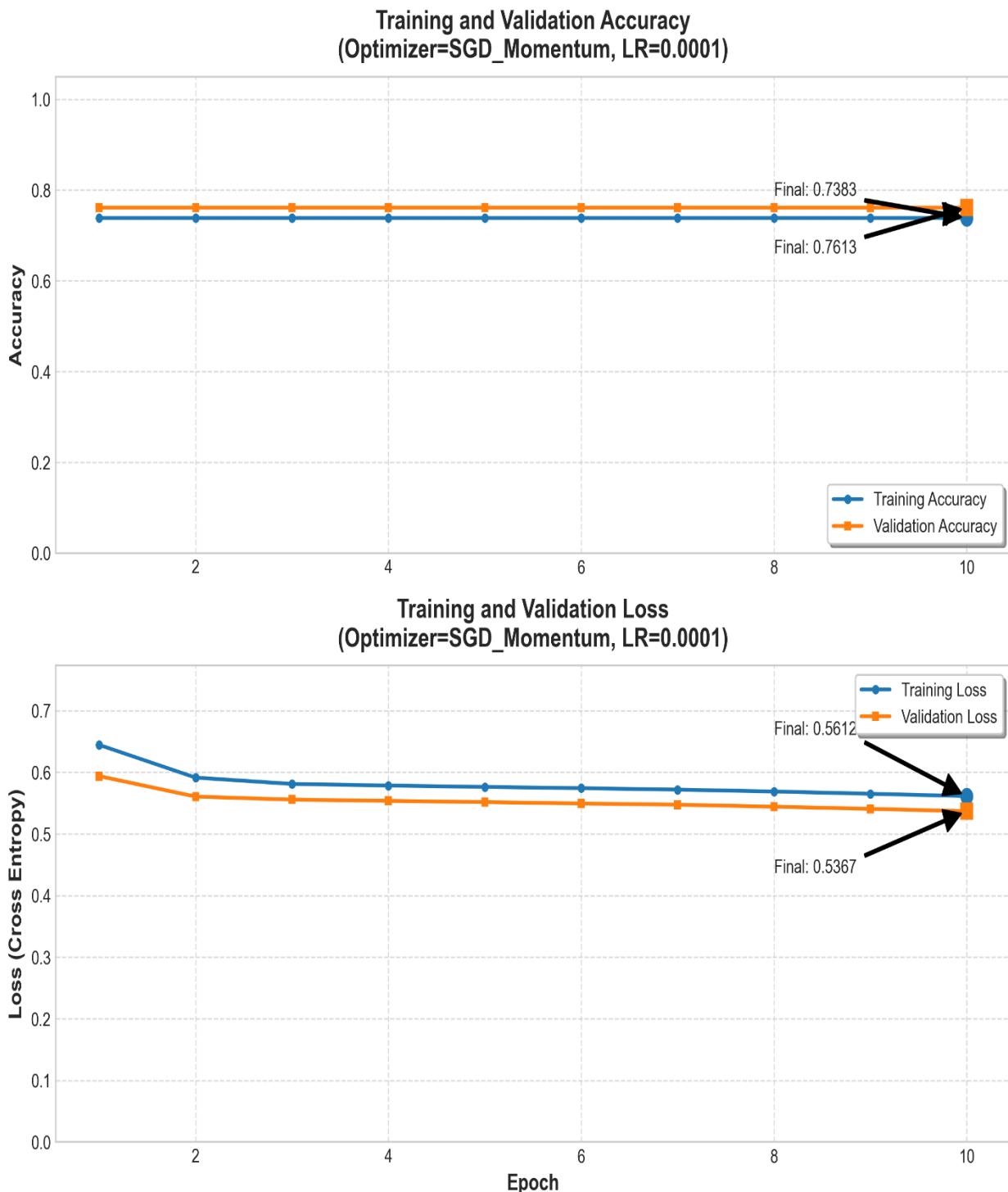
➤ SGD with Momentum - Graph for learning rate 0.0001:

❖ 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

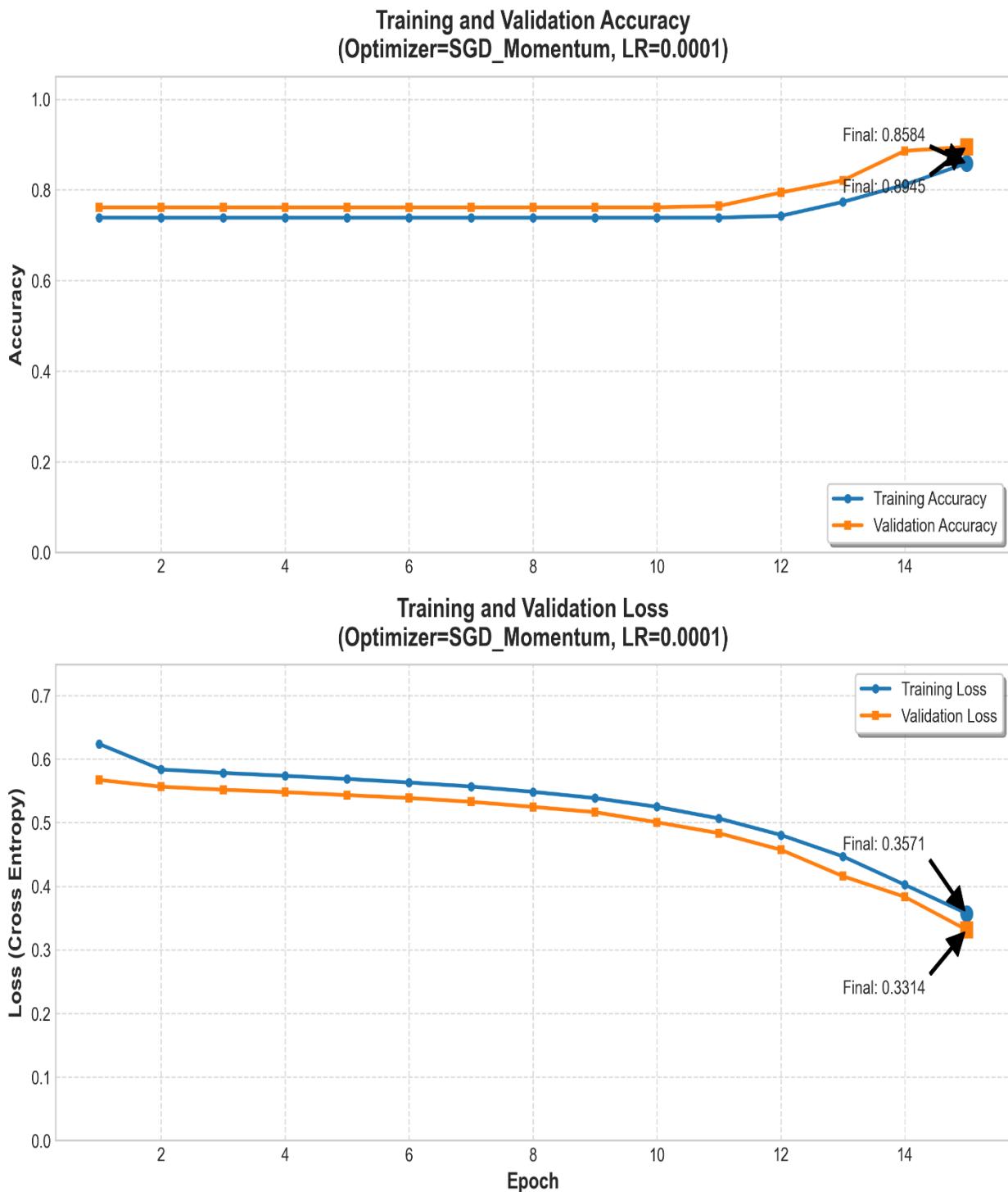
❖ Epochs 10:



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:



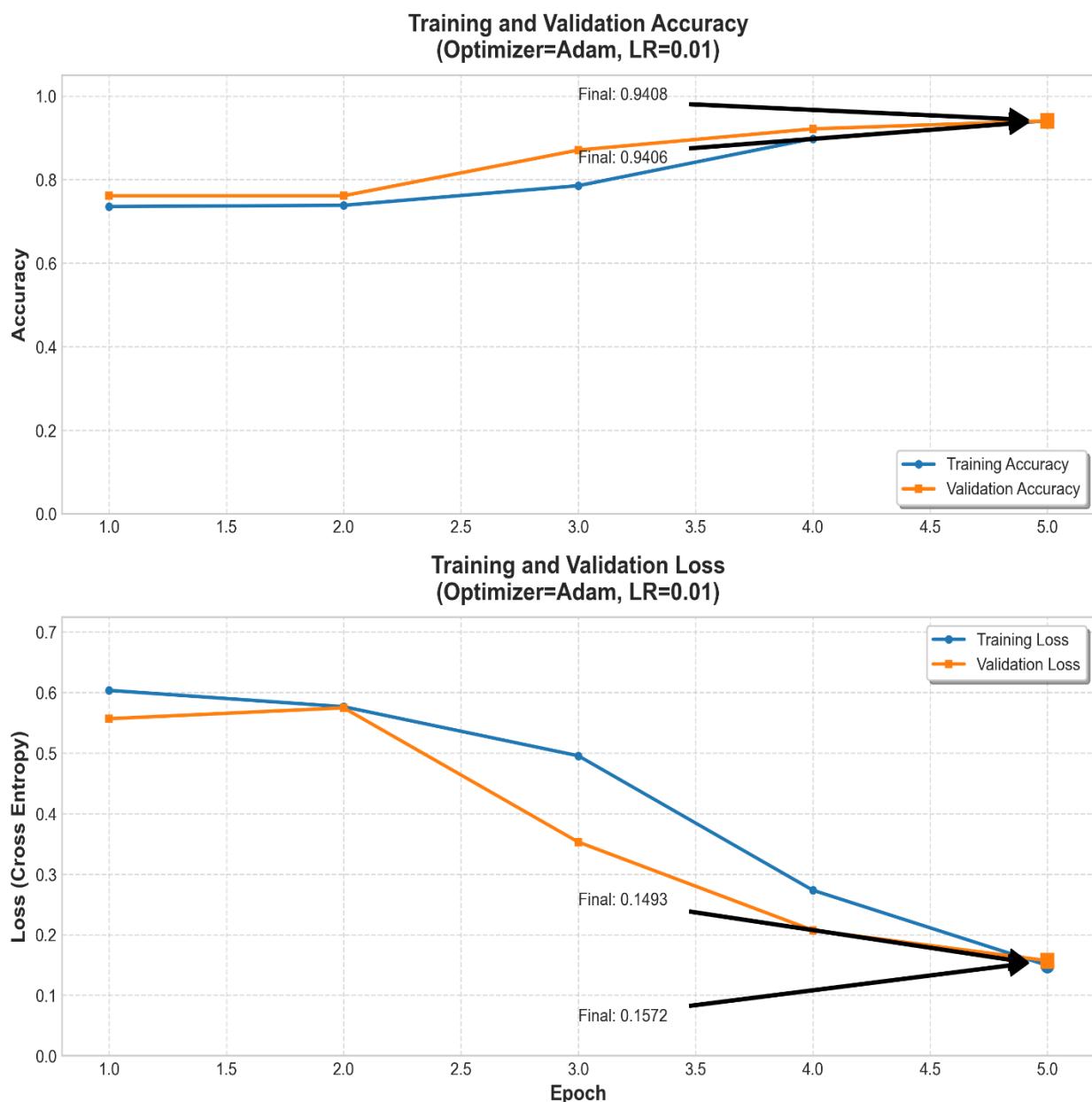
Summary Metrics:
 • Final Training Accuracy: 0.8584, Validation Accuracy: 0.8945
 • Best Validation Accuracy: 0.8945 (Epoch 15)
 • Total Images: 1043 training, 1043 validation

ג. אלגוריתם Adam

אחד האופטימיזרים הפופולריים למיניהם عمוקה – משלב Adaptive Learning rate – משלב Momentum.

➤ Adam - Graph for learning rate 0.01:

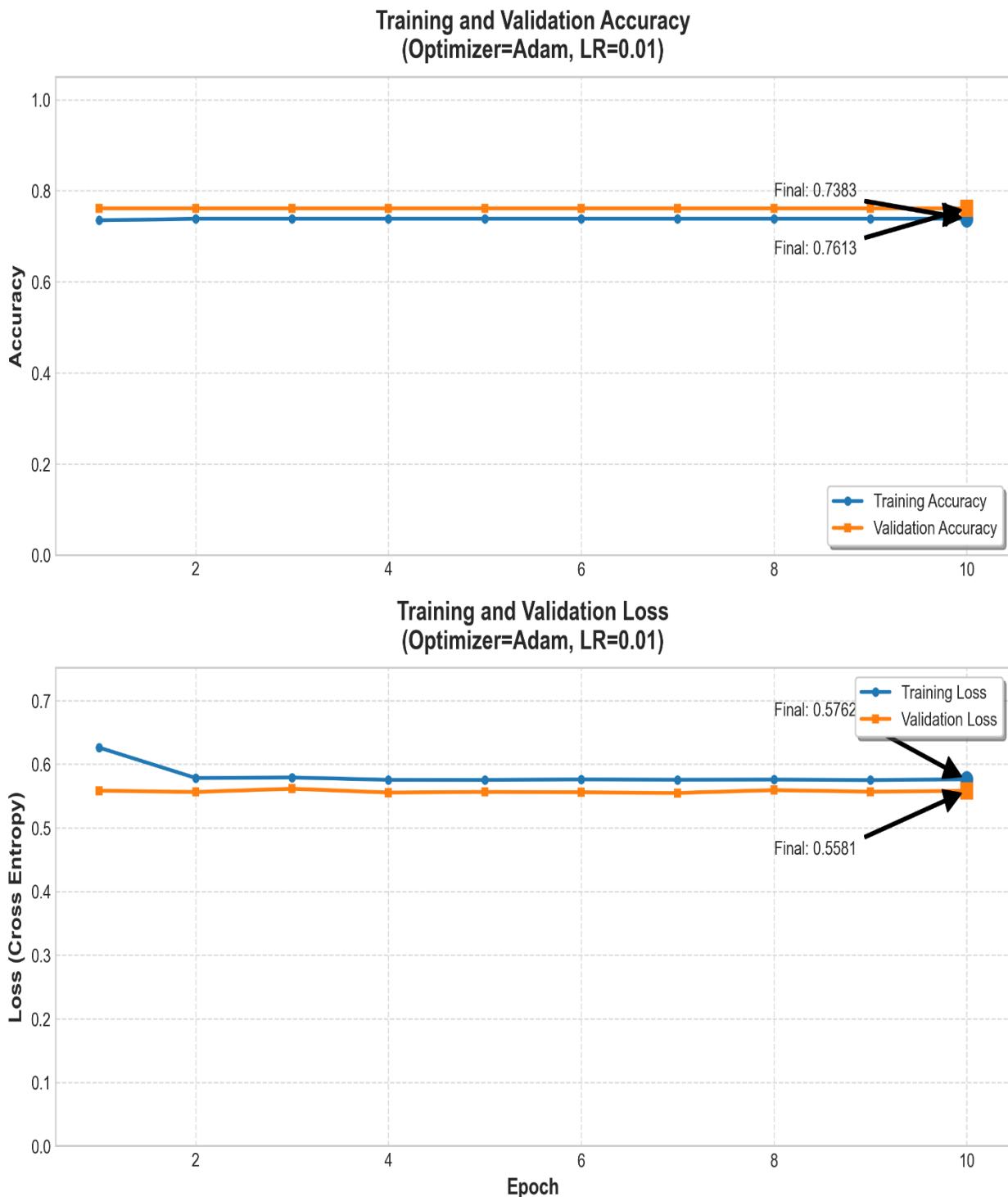
❖ Epochs 5:



Summary Metrics:

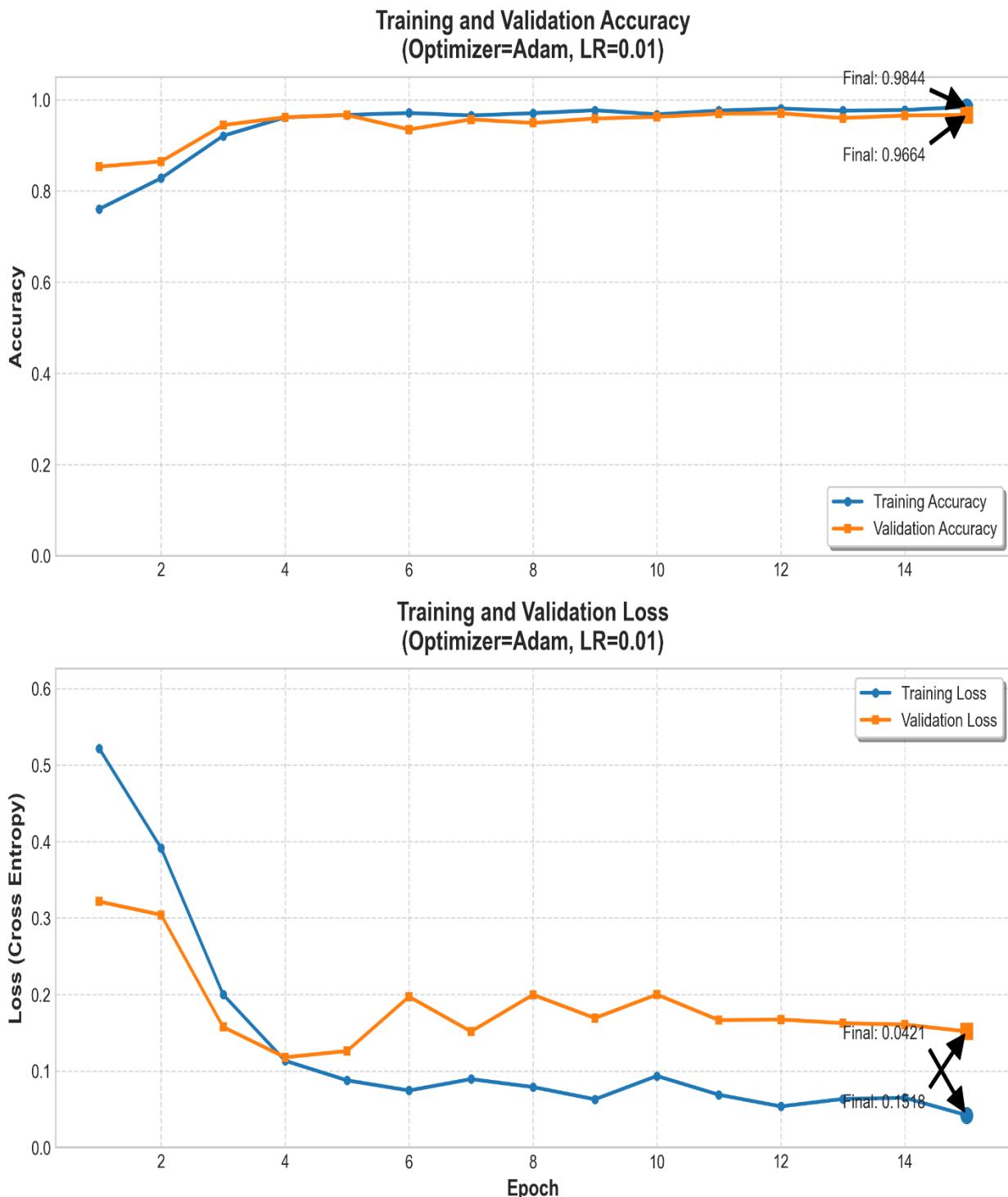
- Final Training Accuracy: 0.9408, Validation Accuracy: 0.9406
- Best Validation Accuracy: 0.9406 (Epoch 5)
- Total Images: 1043 training, 1043 validation

❖ 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

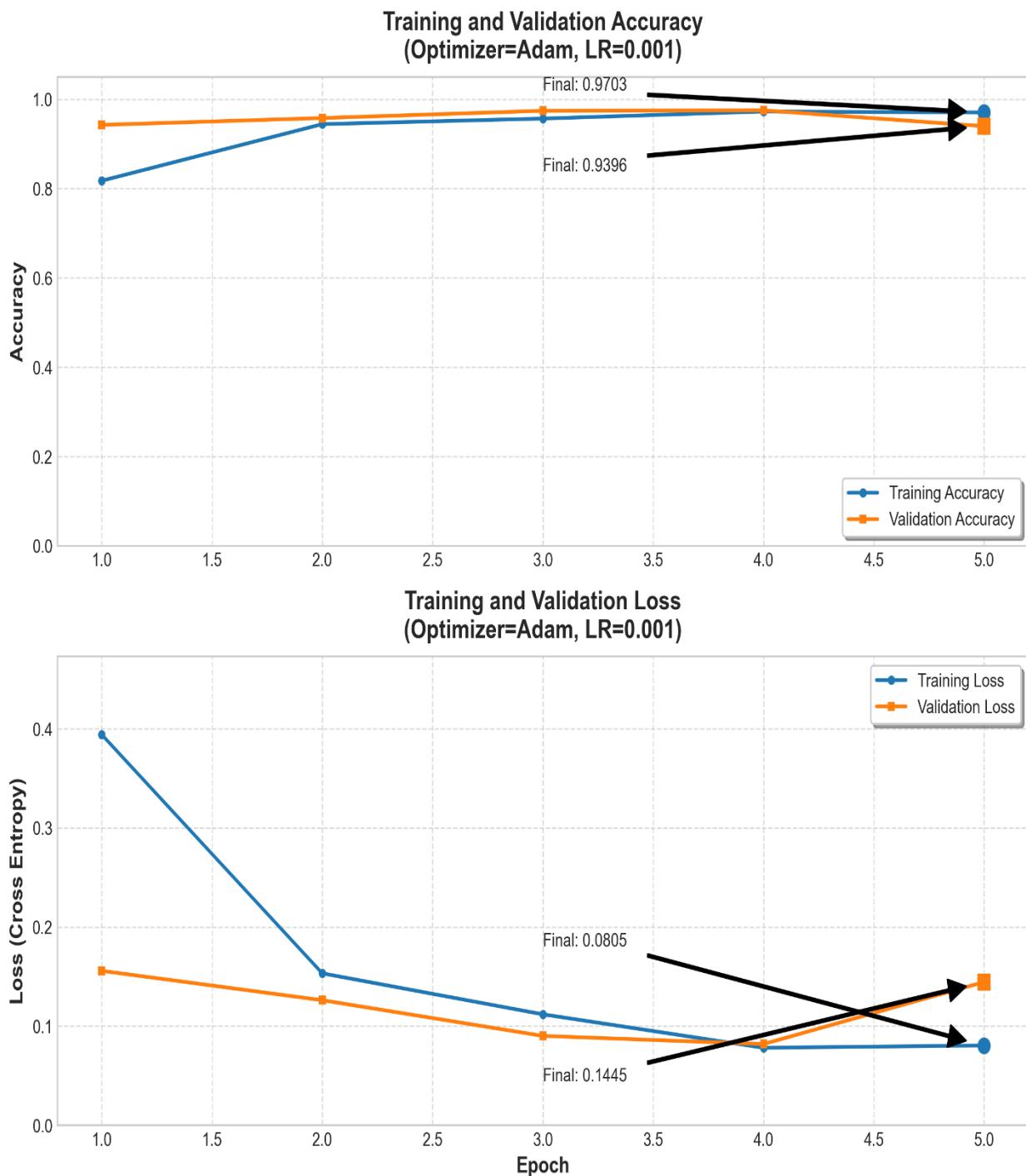
❖ Epochs 15:



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

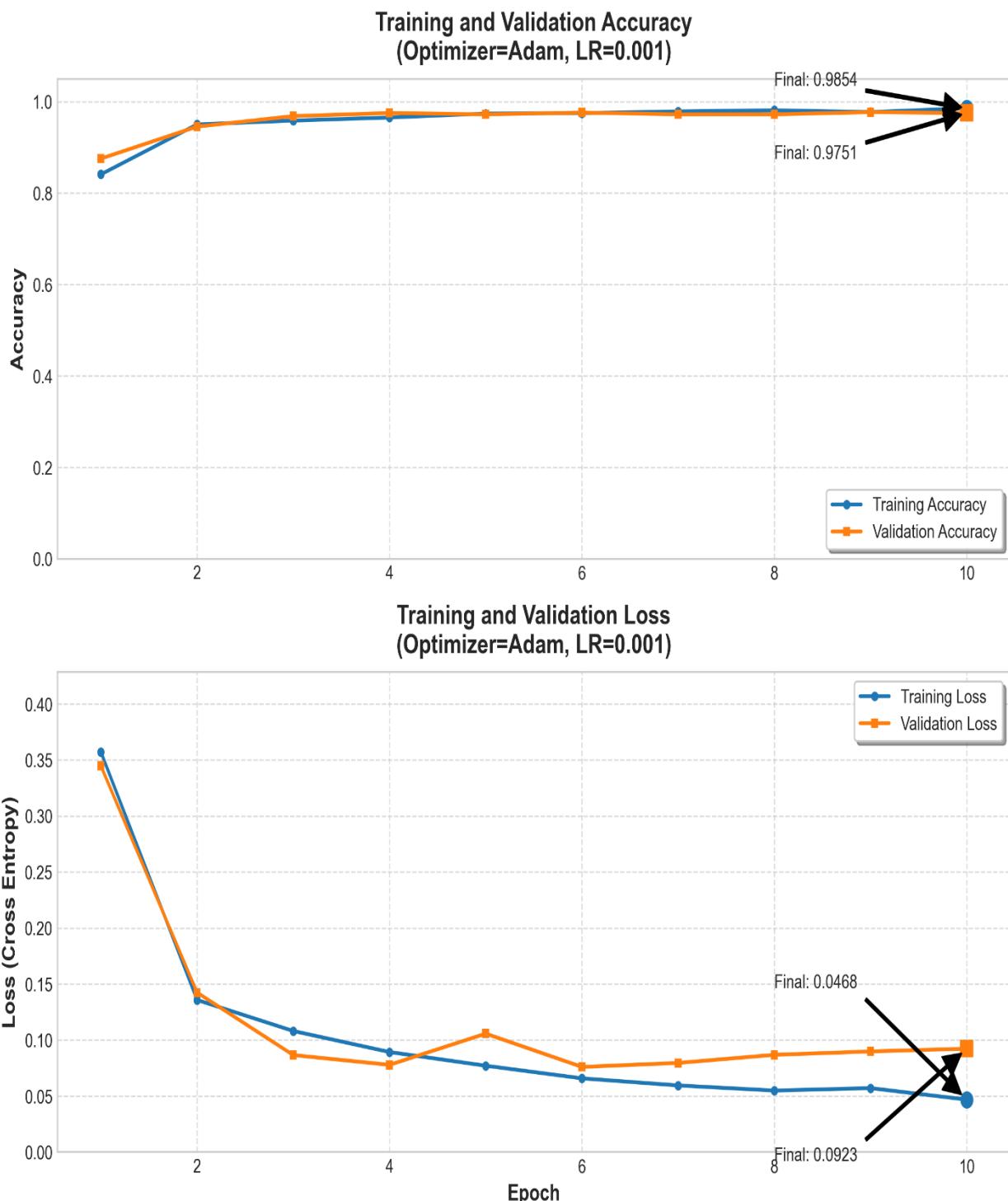
➤ Adam - Graph for learning rate 0.001:

❖ Epochs 5:



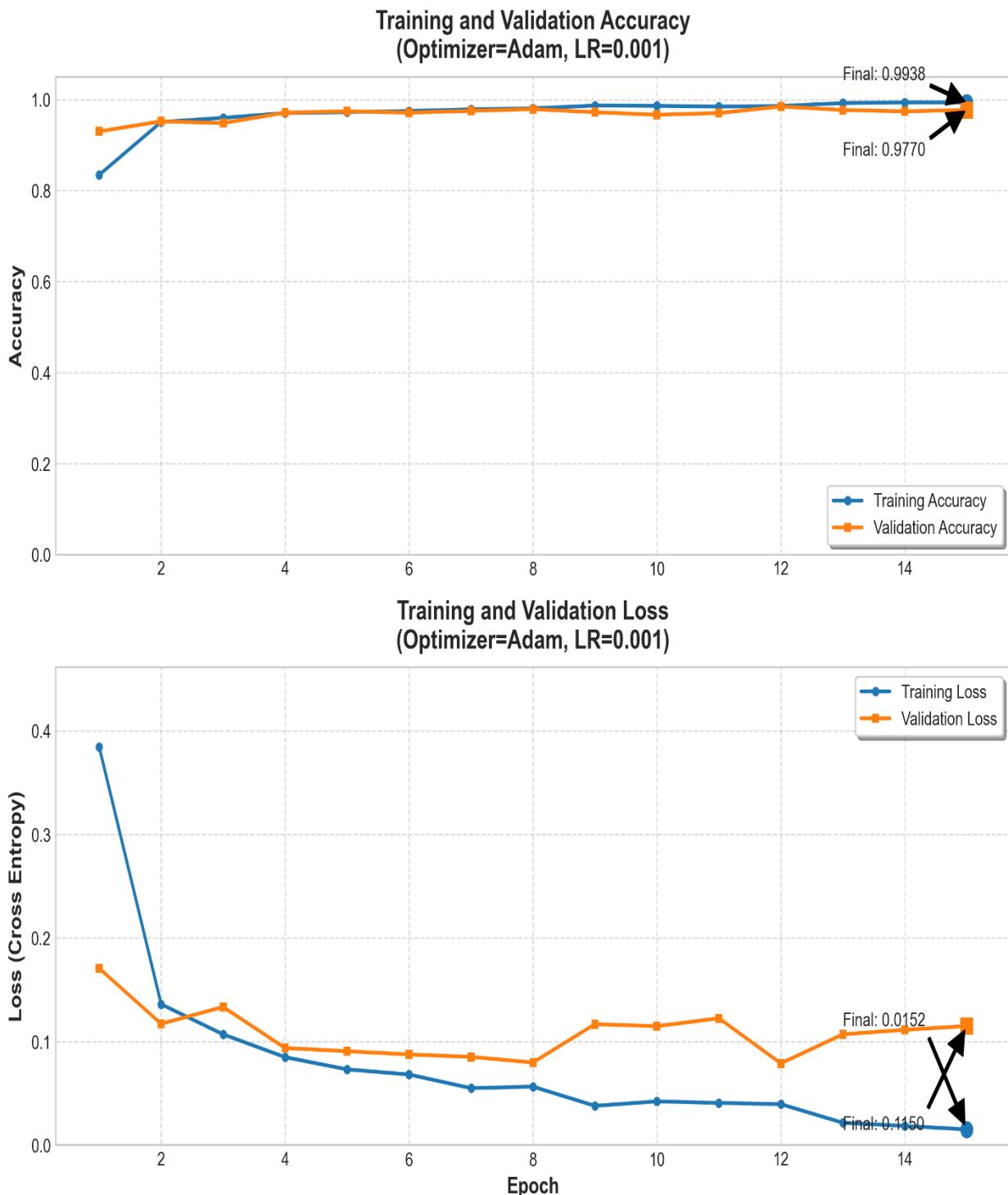
Summary Metrics:
 • Final Training Accuracy: 0.9703, Validation Accuracy: 0.9396
 • Best Validation Accuracy: 0.9751 (Epoch 4)
 • Total Images: 1043 training, 1043 validation

❖ 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

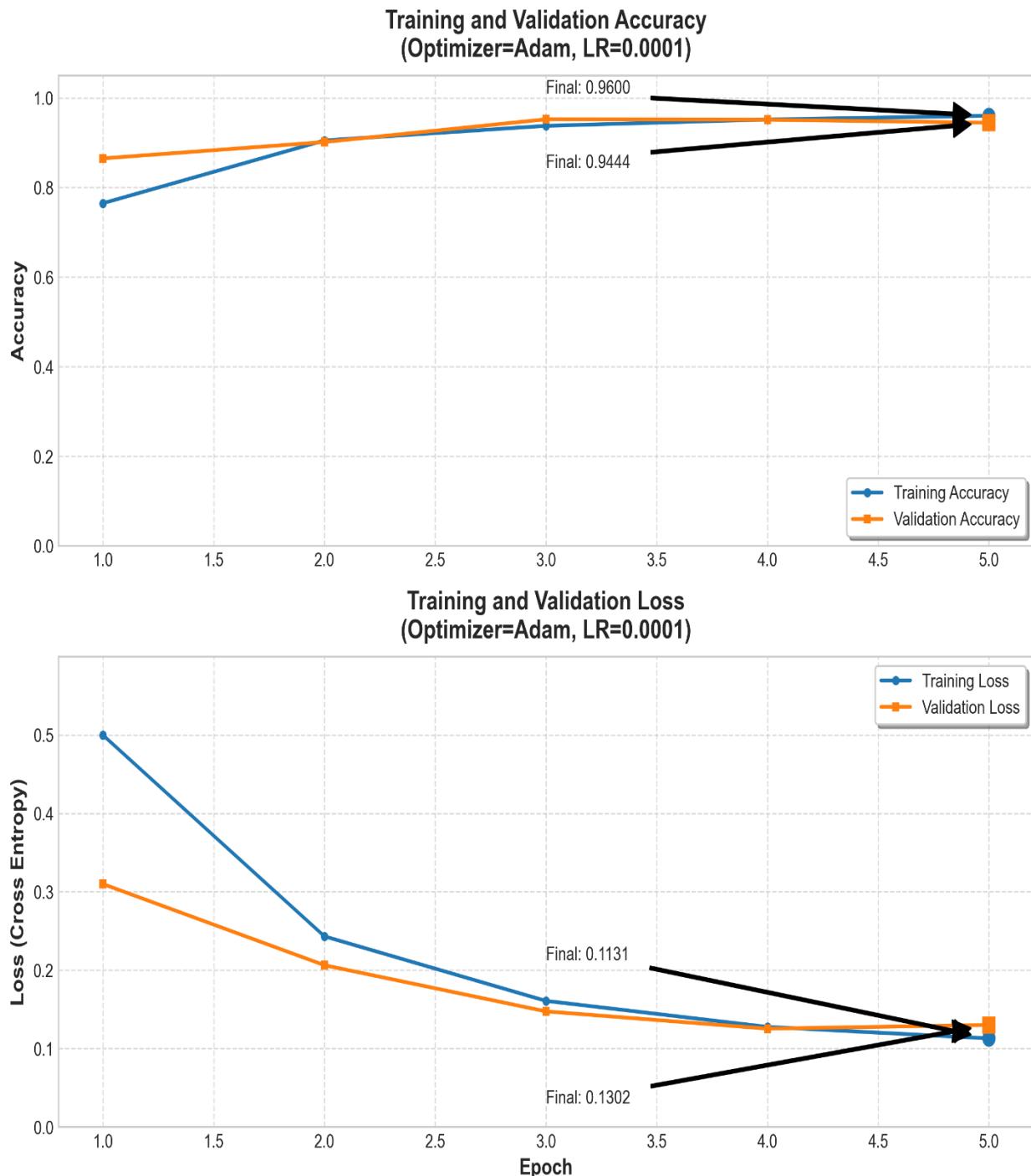
❖ Epochs 15:



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:

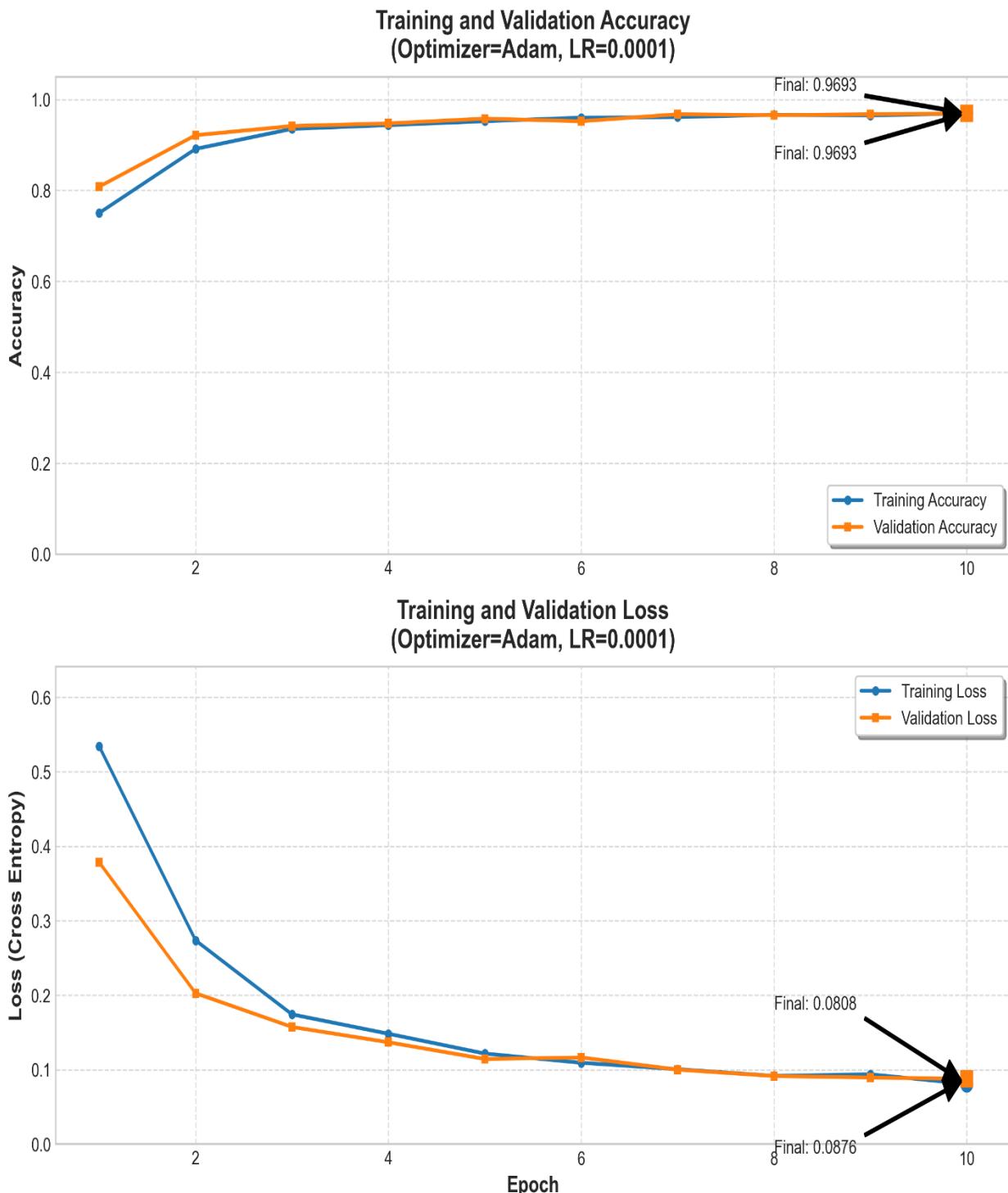
❖ Epochs 5:



Summary Metrics:

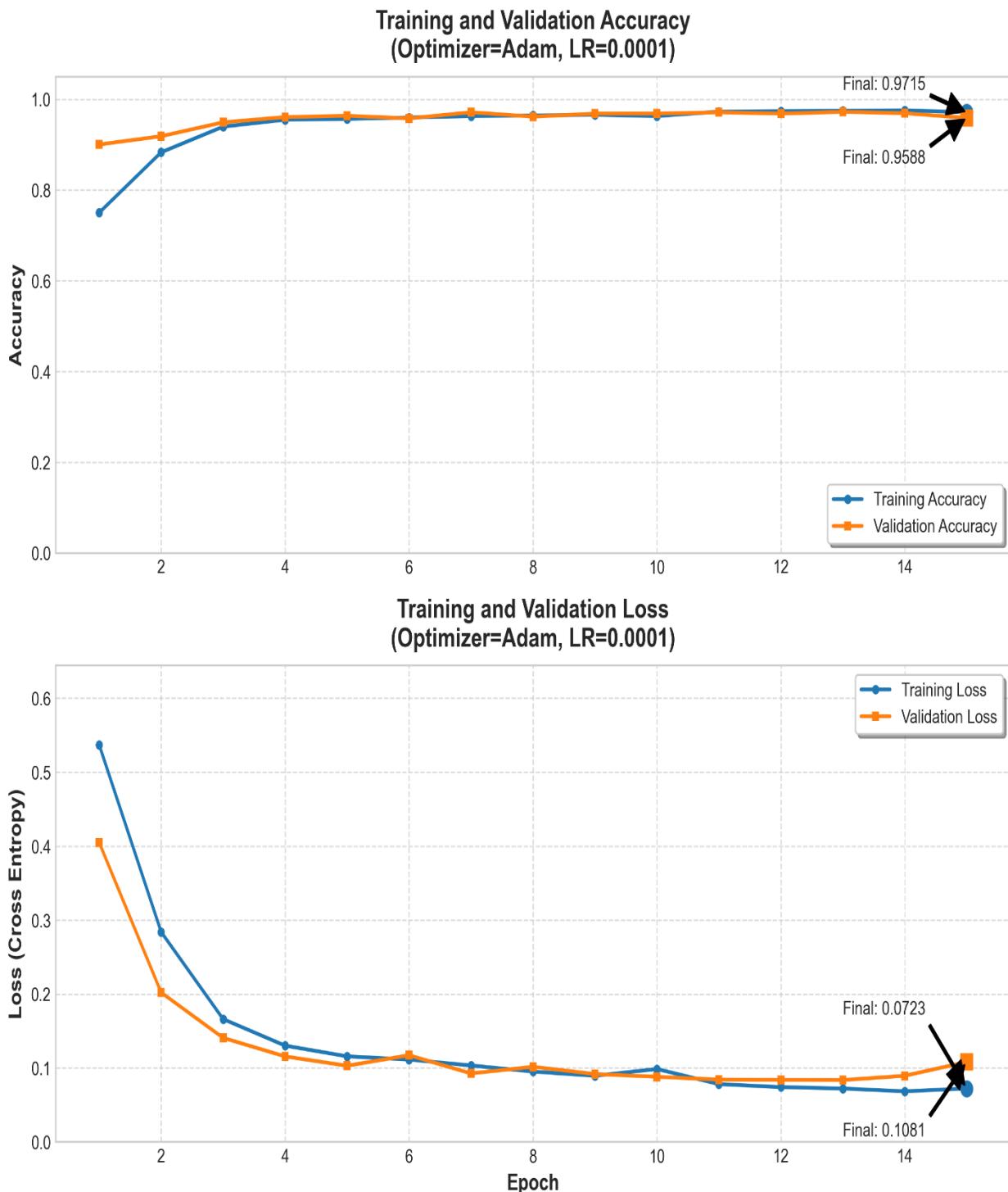
- Final Training Accuracy: 0.9600, Validation Accuracy: 0.9444
- Best Validation Accuracy: 0.9521 (Epoch 3)
- Total Images: 1043 training, 1043 validation

❖ 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:



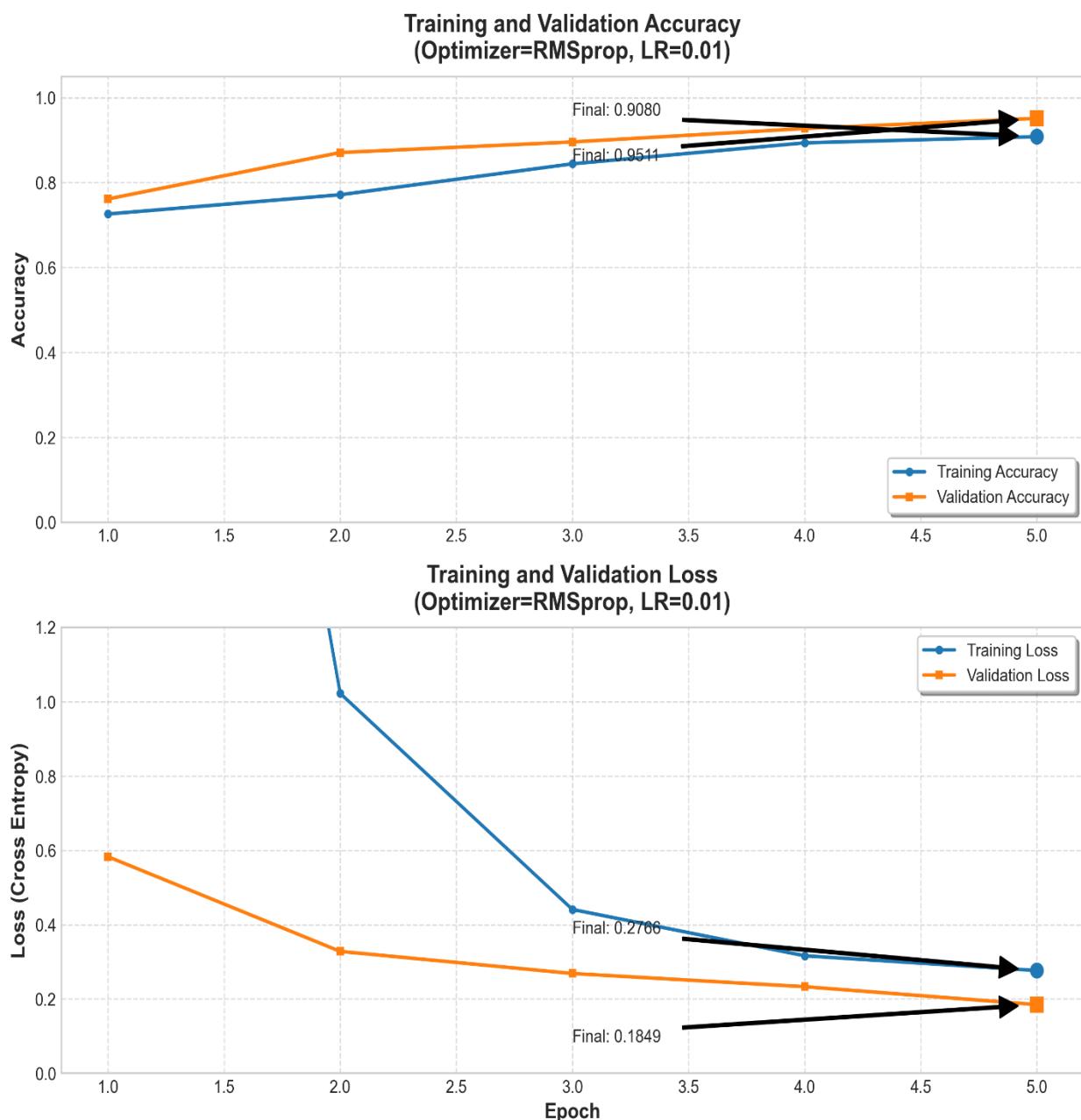
Summary Metrics:
 • Final Training Accuracy: 0.9715, Validation Accuracy: 0.9588
 • Best Validation Accuracy: 0.9722 (Epoch 13)
 • Total Images: 1043 training, 1043 validation

ד. אלגוריתם RMSprop

אופטימיזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

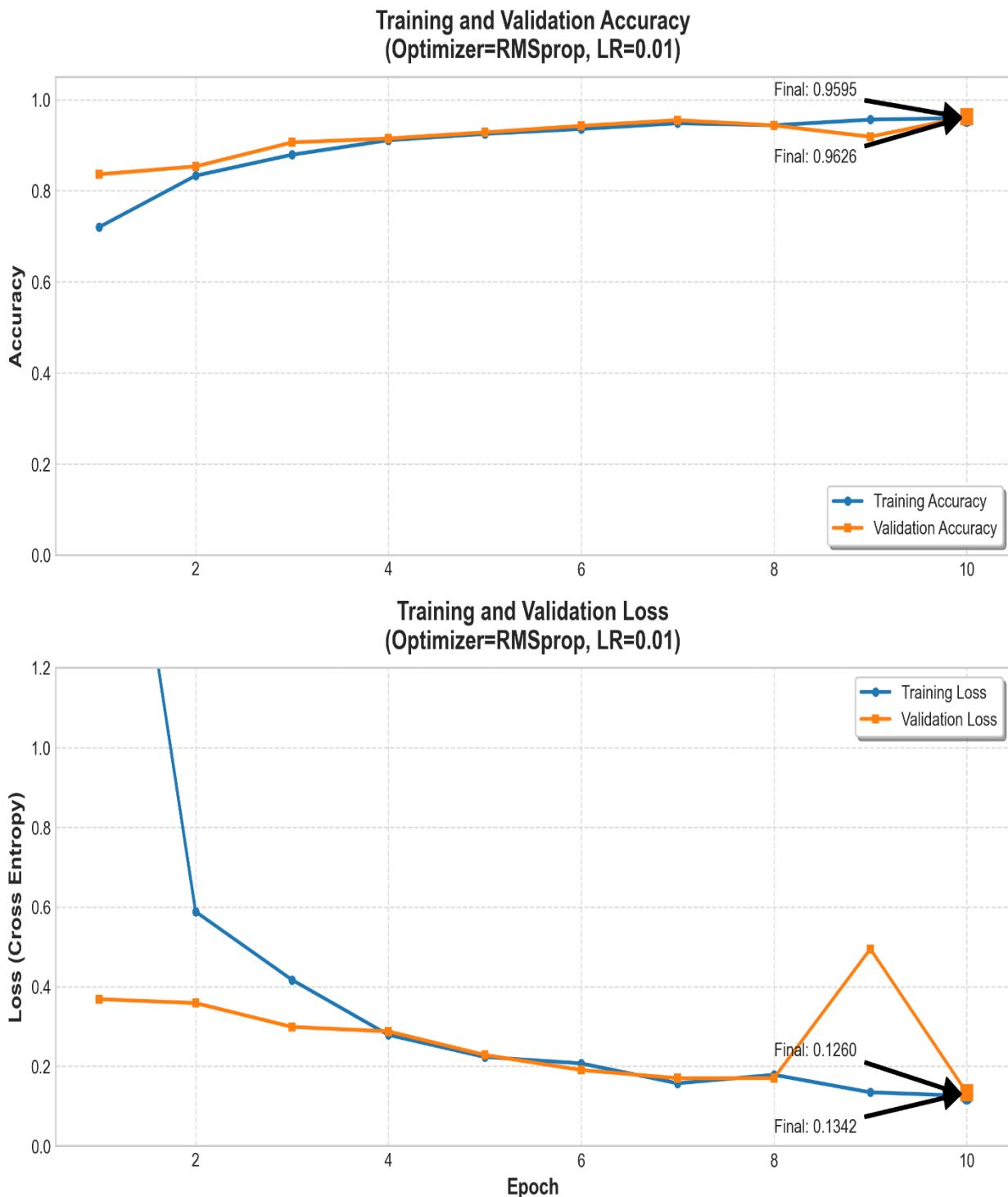
➤ RMSprop - Graph for learning rate 0.01:

❖ Epochs 5:



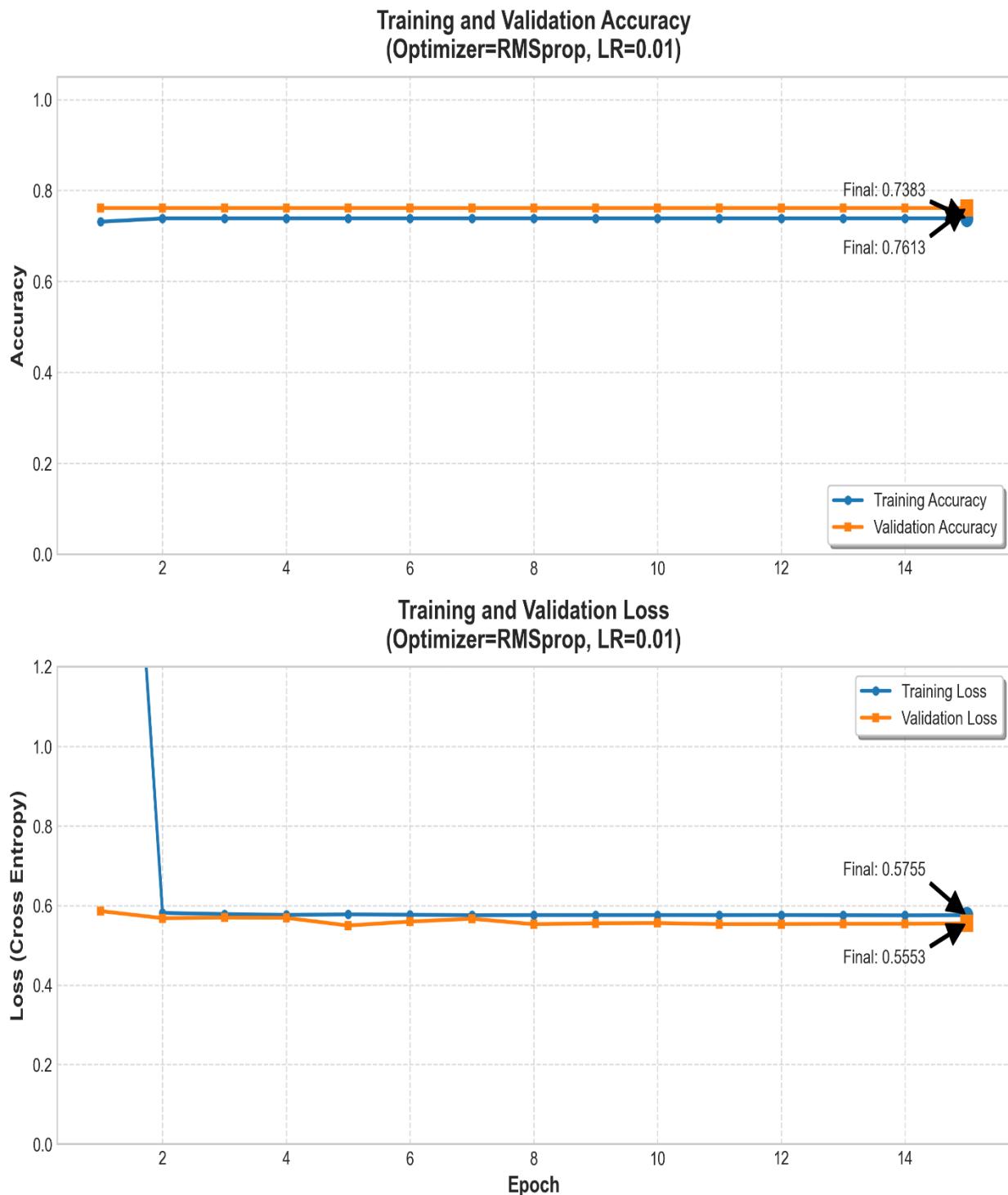
Summary Metrics:
 • Final Training Accuracy: 0.9080, Validation Accuracy: 0.9511
 • Best Validation Accuracy: 0.9511 (Epoch 5)
 • Total Images: 1043 training, 1043 validation

❖ 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

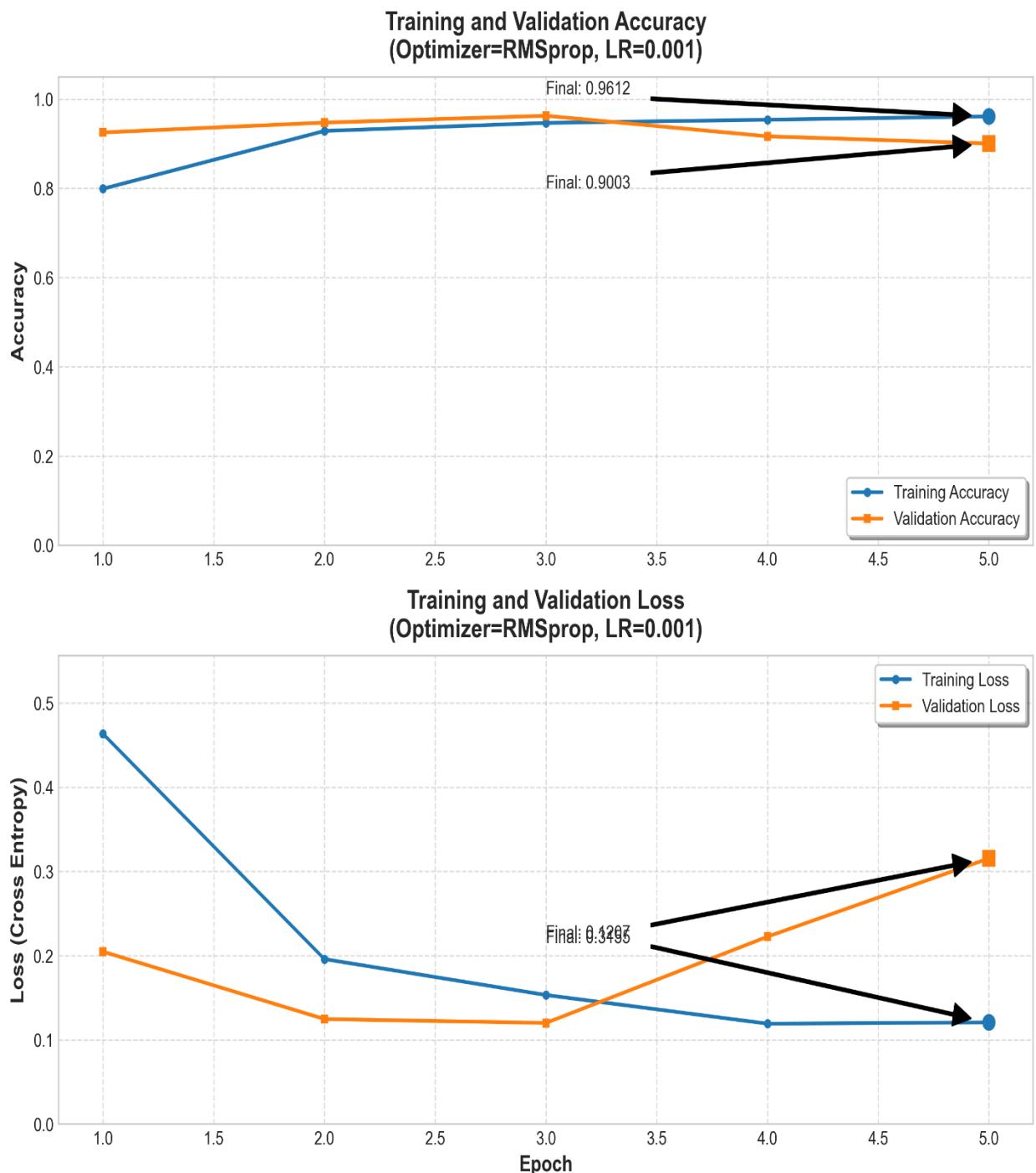
❖ Epochs 15:



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

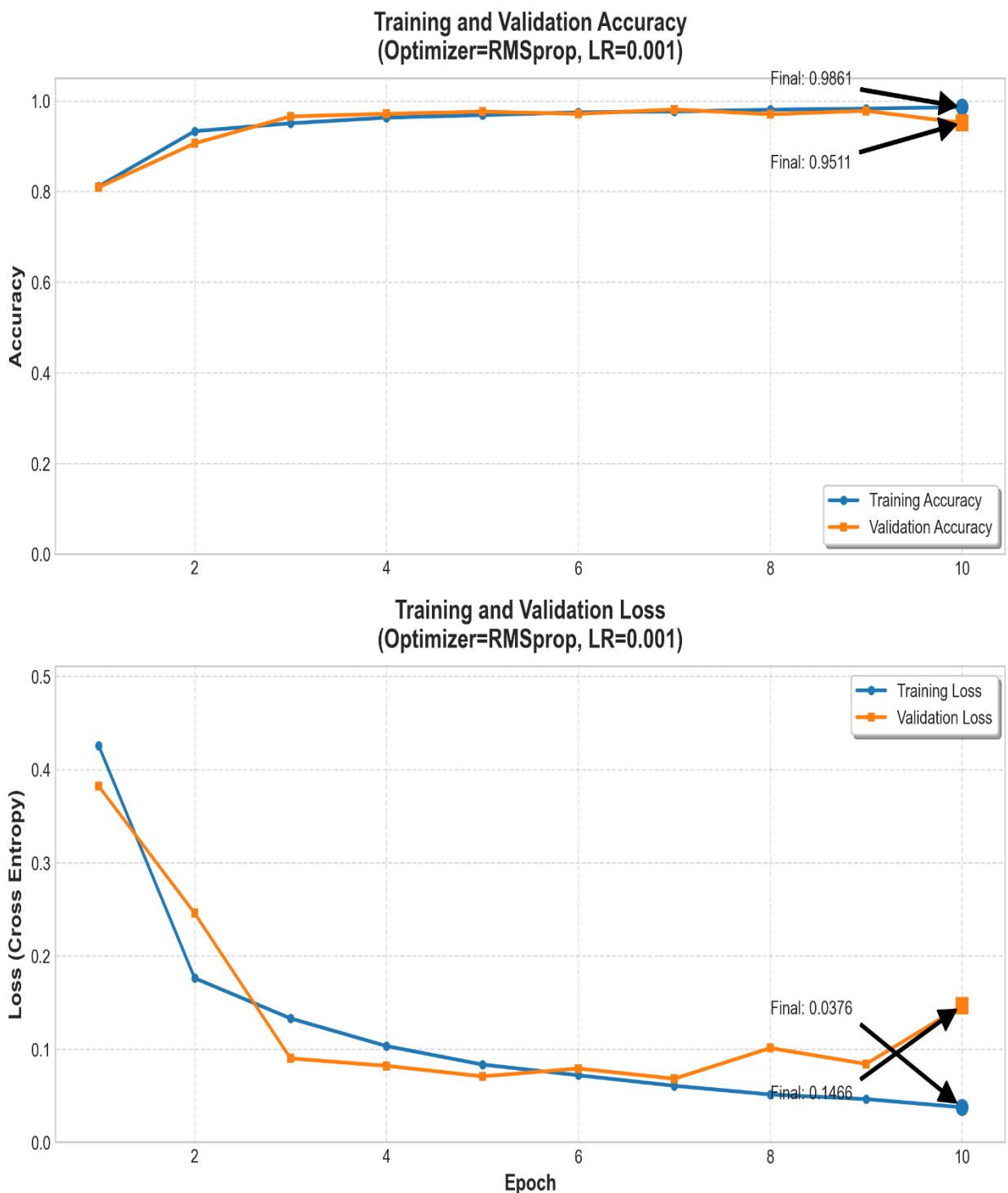
➤ RMSprop - Graph for learning rate 0.001:

❖ Epochs 5:



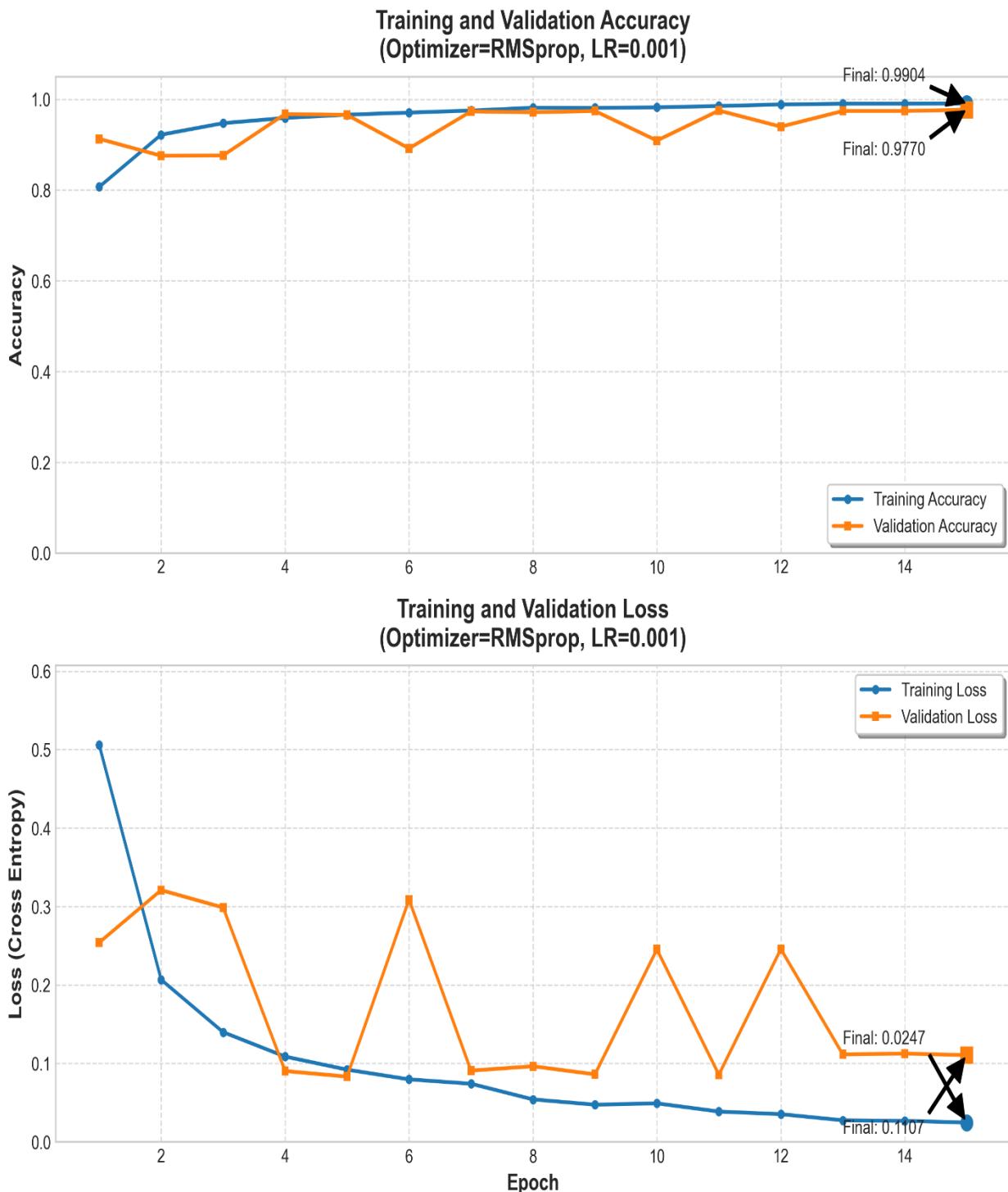
Summary Metrics:
 • Final Training Accuracy: 0.9612, Validation Accuracy: 0.9003
 • Best Validation Accuracy: 0.9626 (Epoch 3)
 • Total Images: 1043 training, 1043 validation

❖ 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

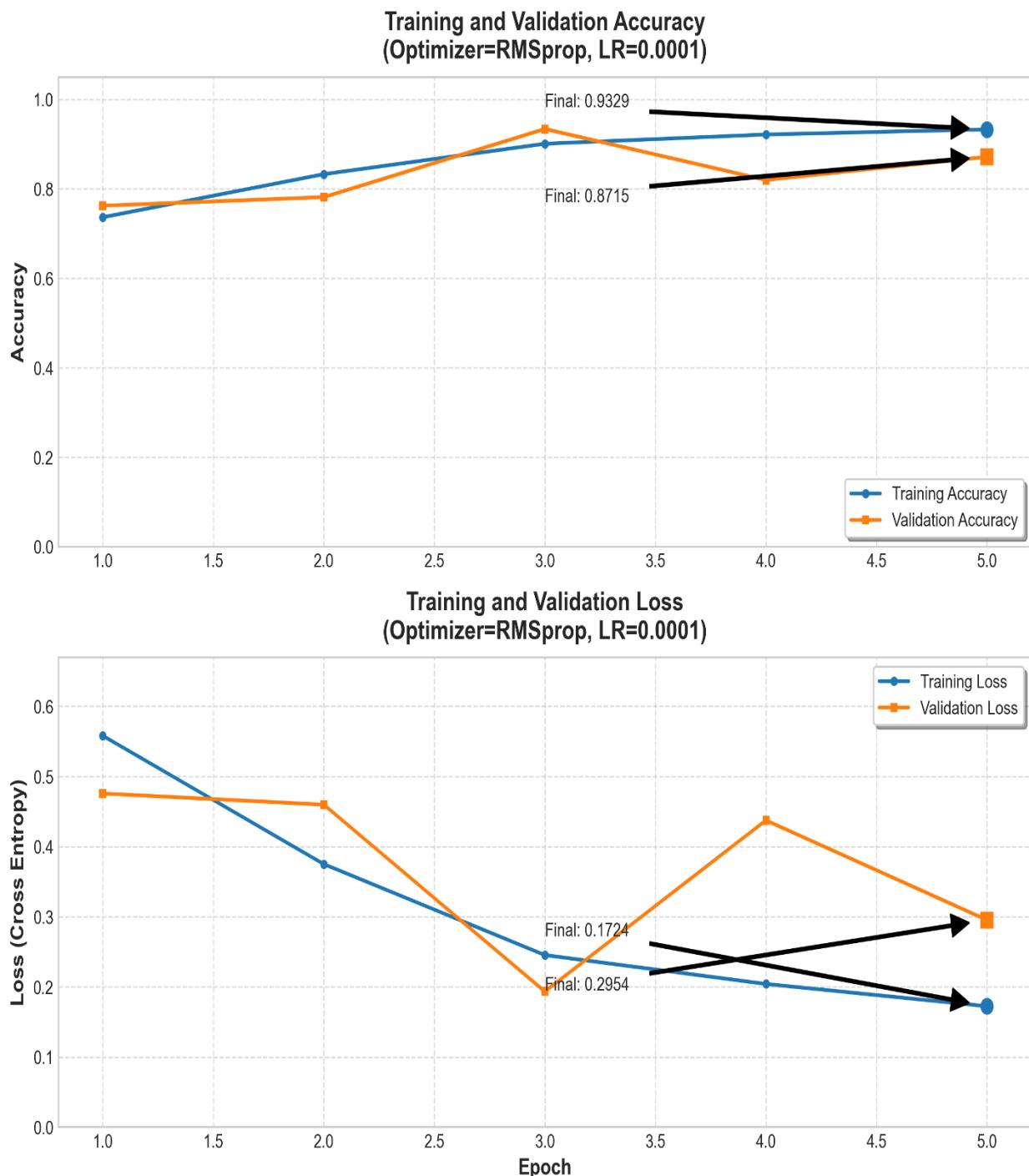
❖ Epochs 15:



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:

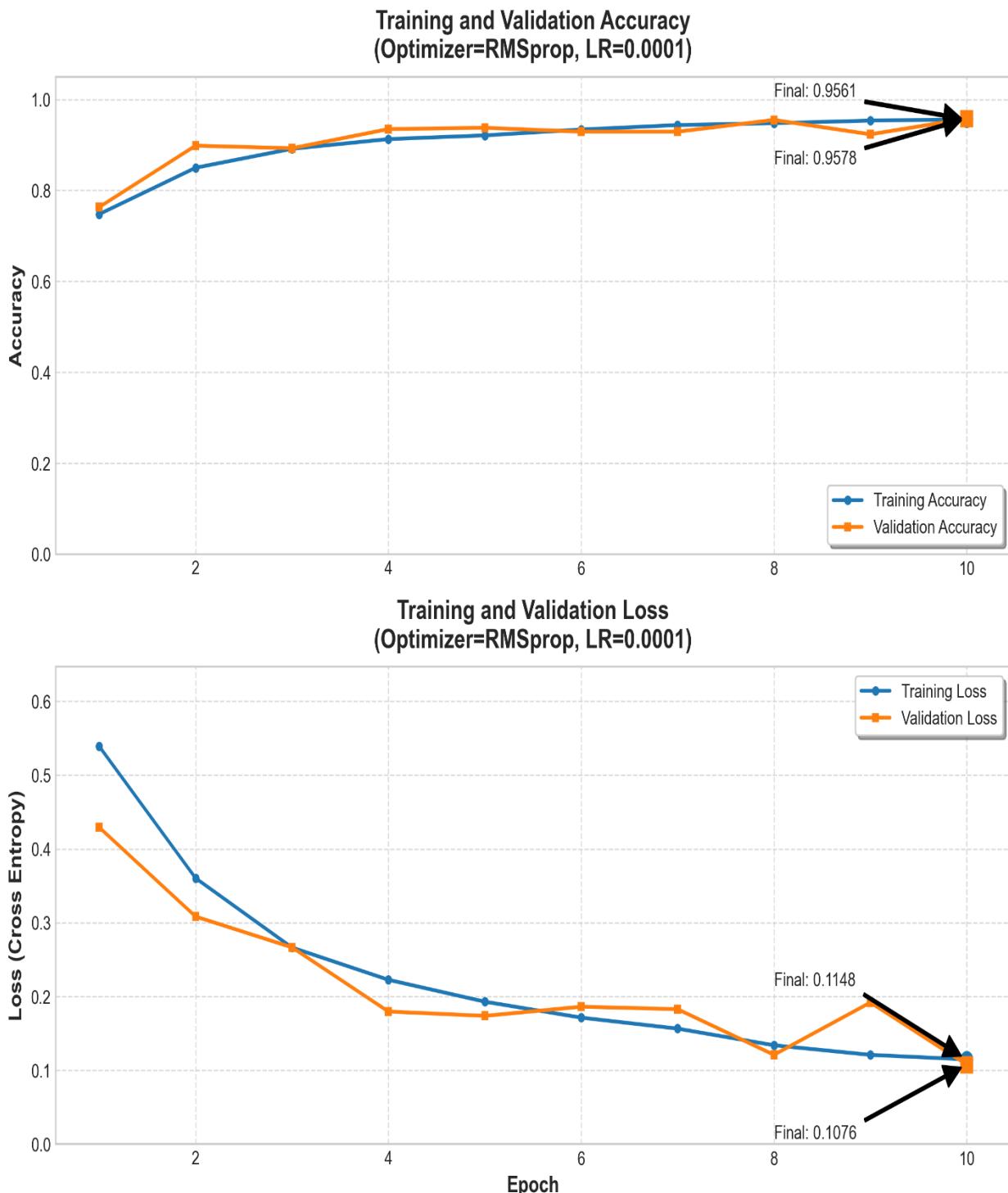
❖ Epochs 5:



Summary Metrics:

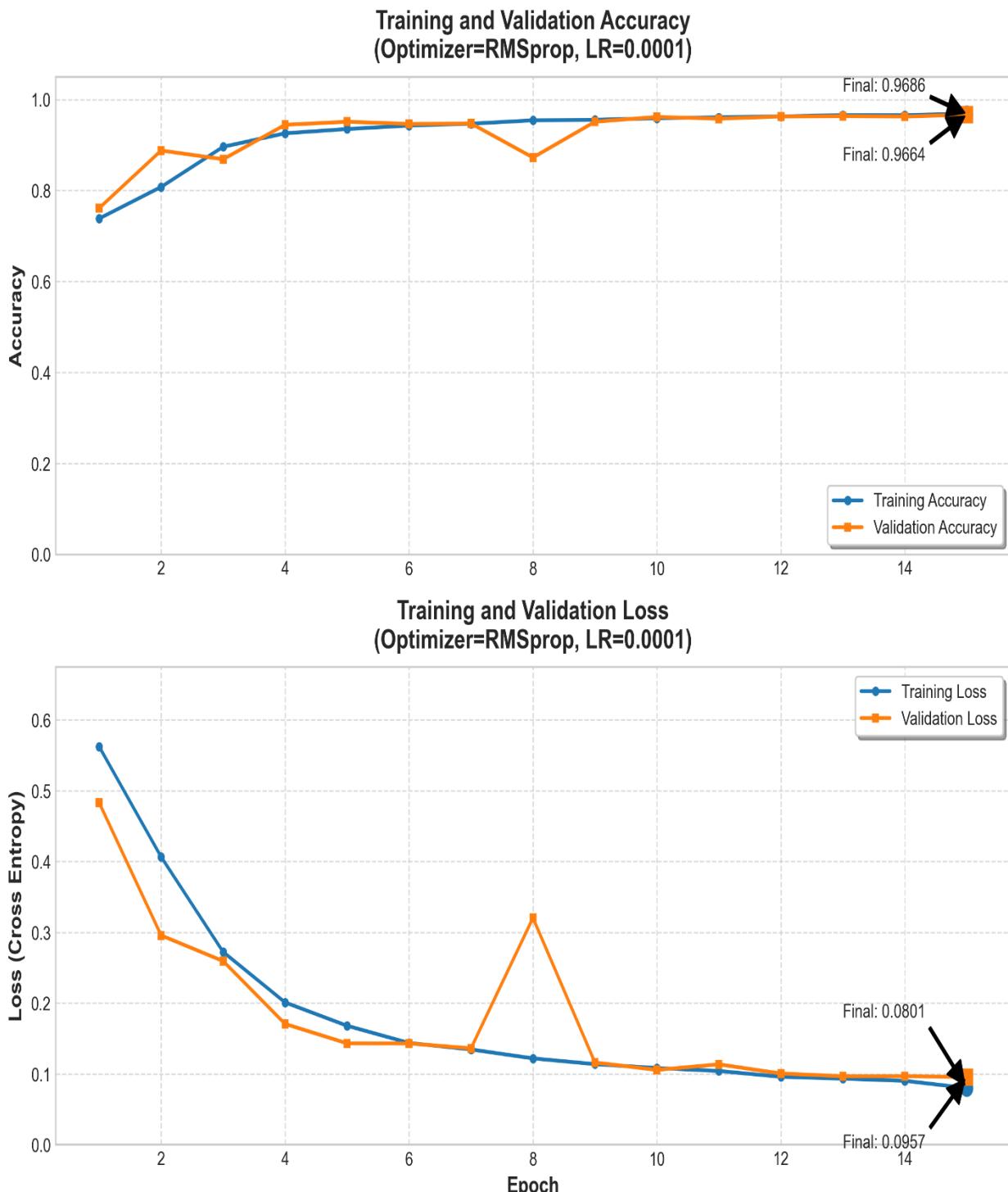
- Final Training Accuracy: 0.9329, Validation Accuracy: 0.8715
- Best Validation Accuracy: 0.9338 (Epoch 3)
- Total Images: 1043 training, 1043 validation

❖ 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

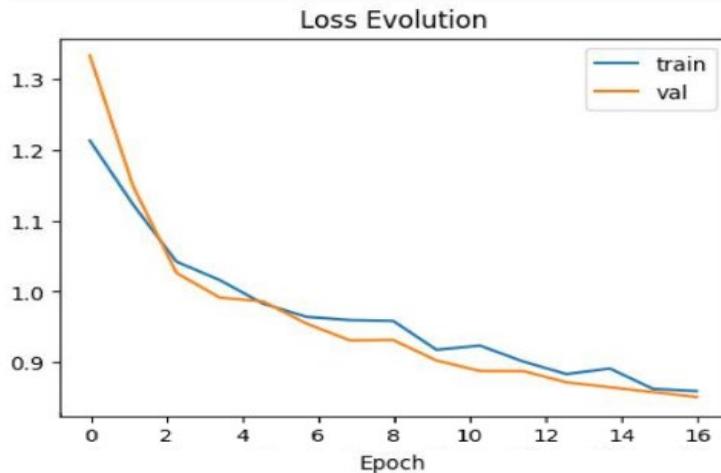
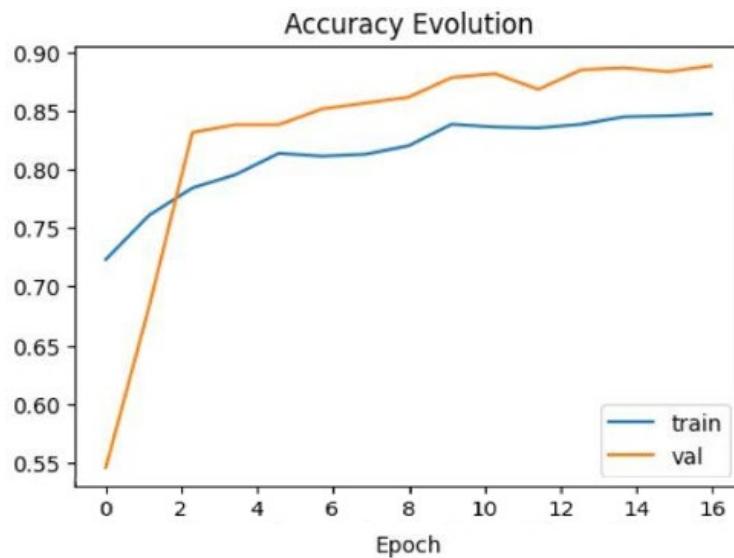
רשות Transfer learning Fine Tuned עם CNN

a. אלגוריתם SGD

בדקנו את השפעת SGD הקלasicי ללא

➤ SGD - Graph for learning rate 0.01:

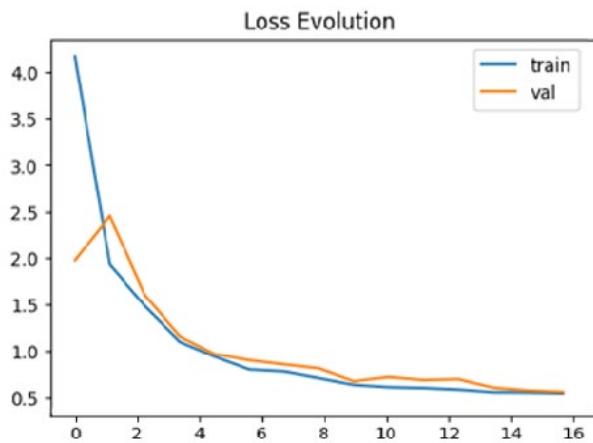
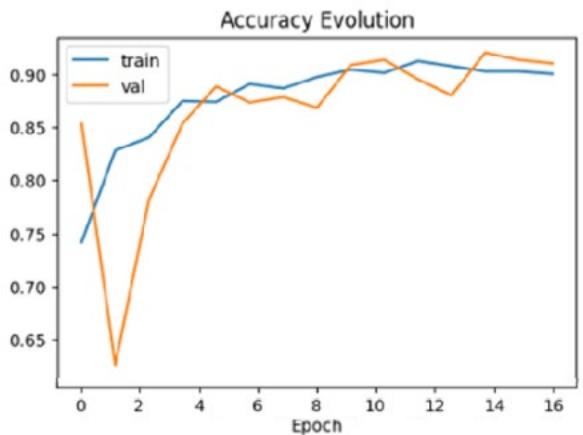
❖ Epochs 17:



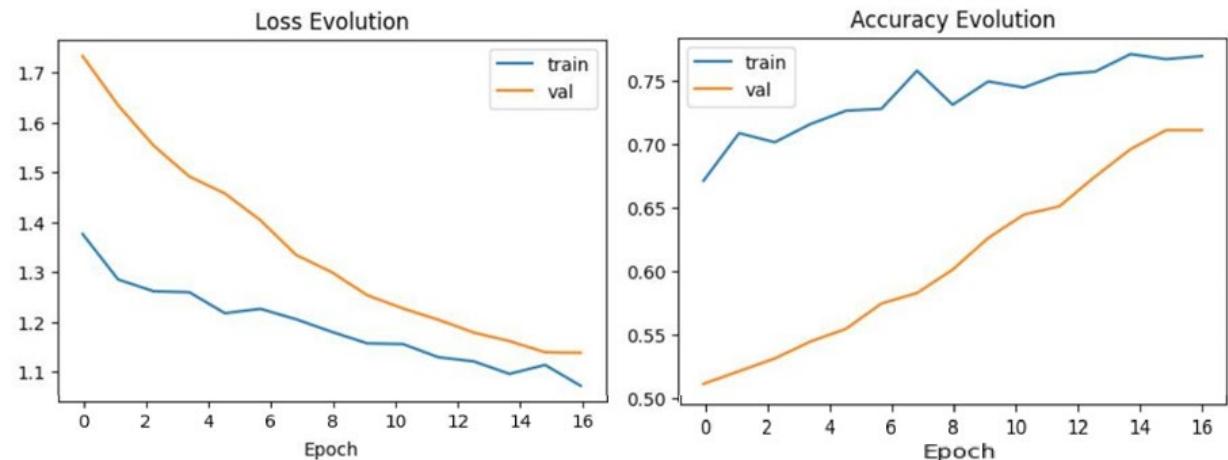
Test accuracy = 89.16%

➤ SGD - Graph for learning rate 0.001:

❖ Epochs 17:



- Test accuracy = 93.33%
- SGD - Graph for learning rate 0.0001:
- ❖ Epochs 17:



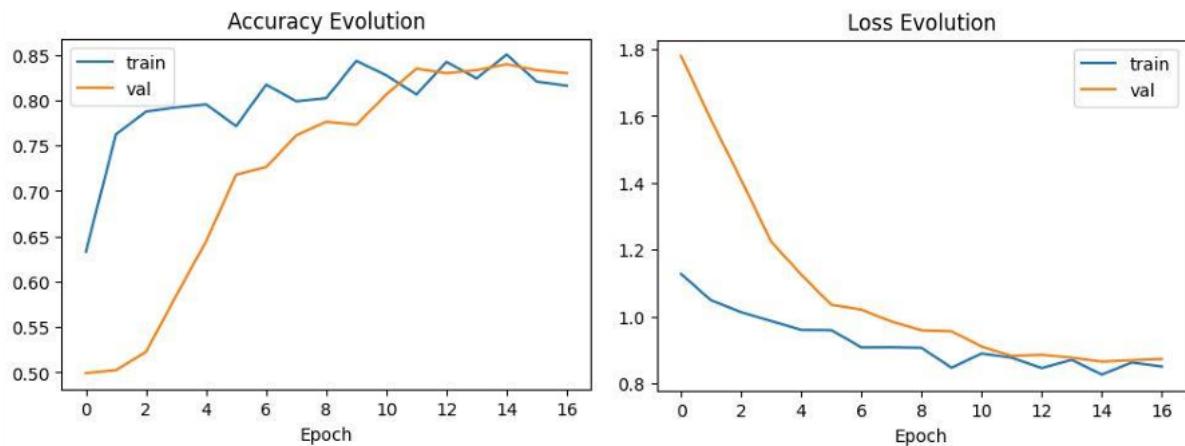
Test accuracy = 72.66%

ב. אלגוריתם SGD עם Momentum

Nousfeh תמייה בפרמטר $\eta = 0.9$

- SGD with Momentum - Graph for learning rate 0.01:

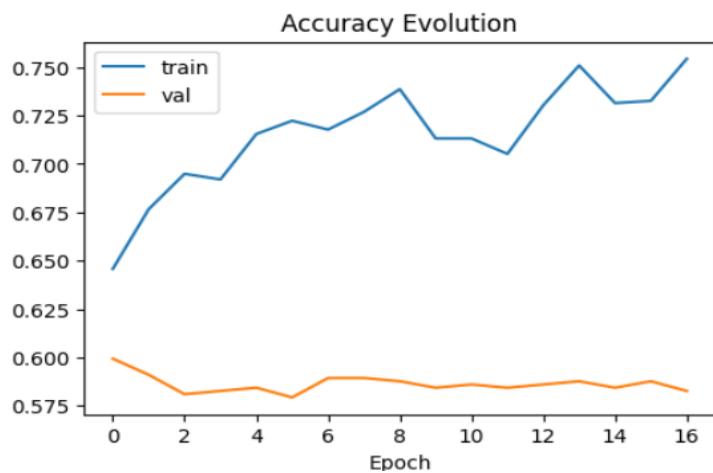
- ❖ Epochs 17:

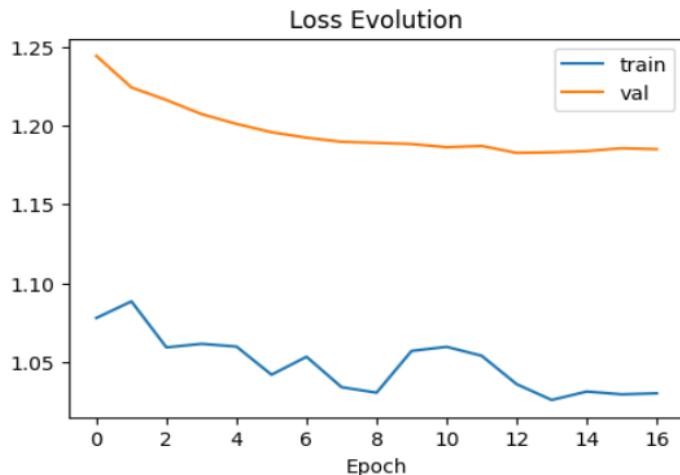


Test accuracy = 84.83%

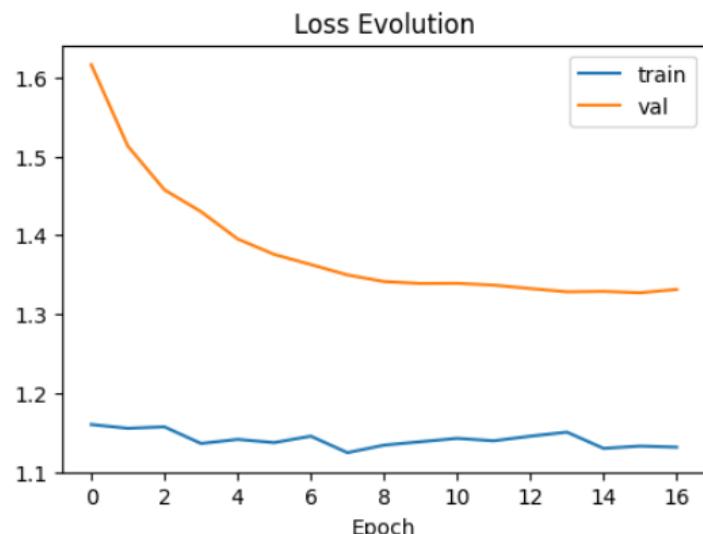
- SGD with Momentum - Graph for learning rate 0.001:

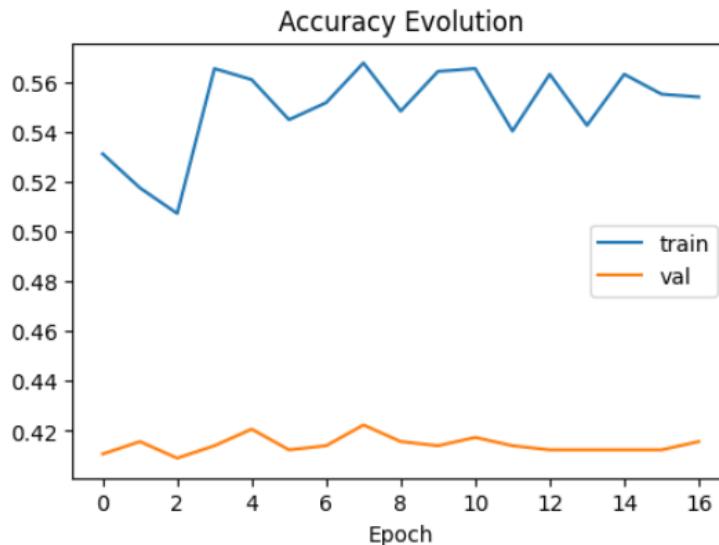
- ❖ Epochs 17:





- Test accuracy = 56.66%
- SGD with Momentum - Graph for learning rate 0.0001:
- ❖ Epochs 17:





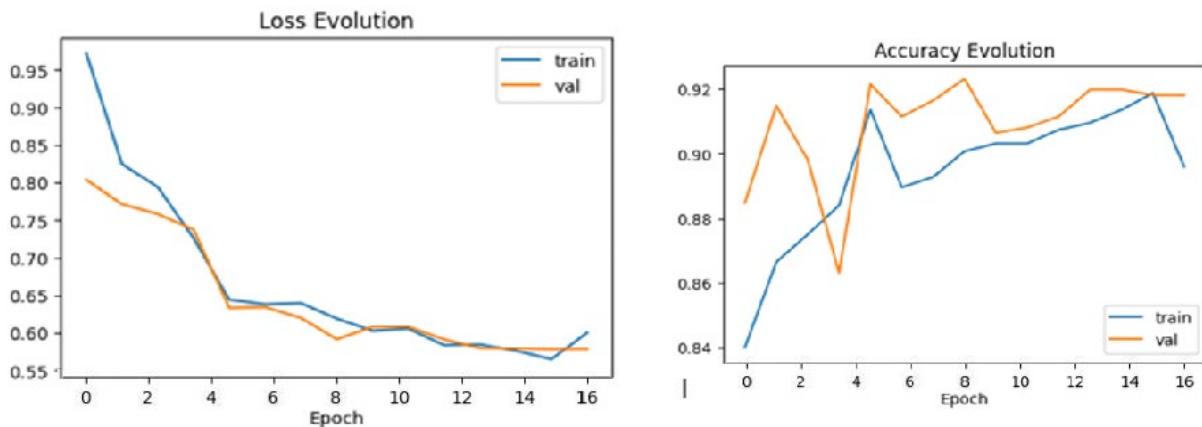
Test accuracy = 42.33%

ג. אלגוריתם Adam

- Adaptive Learning – משלב Momentum

➤ Adam - Graph for learning rate 0.01:

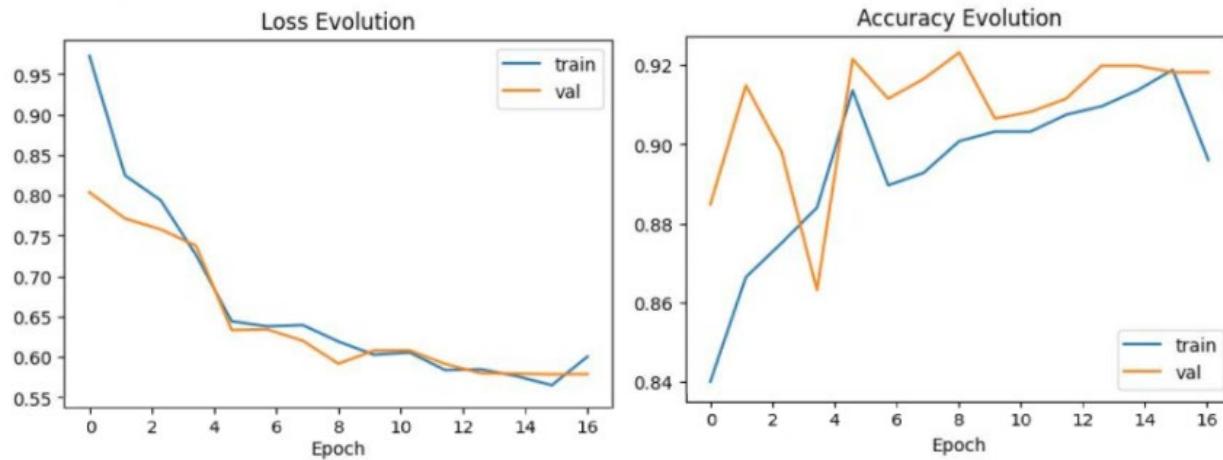
❖ Epochs 17:



Test accuracy = 92.66%

➤ Adam - Graph for learning rate 0.001:

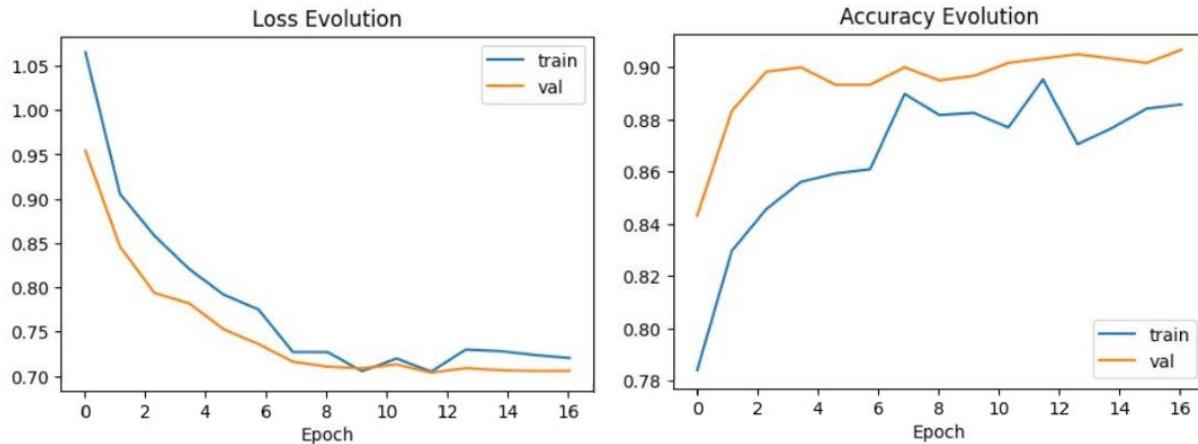
❖ Epochs 17:



➤ Test accuracy = 92.76%

➤ Adam - Graph for learning rate 0.0001:

❖ Epochs 17:



Test accuracy = 91.16%

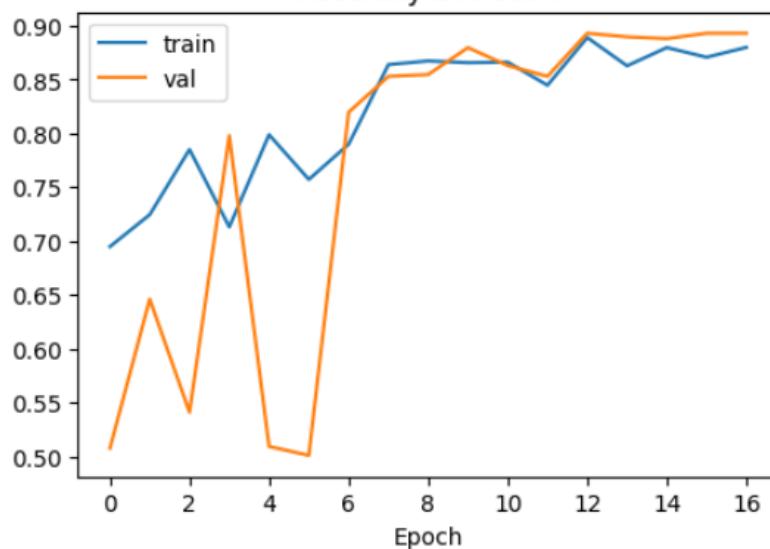
ד. אלגוריתם RMSprop

אופטימיזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

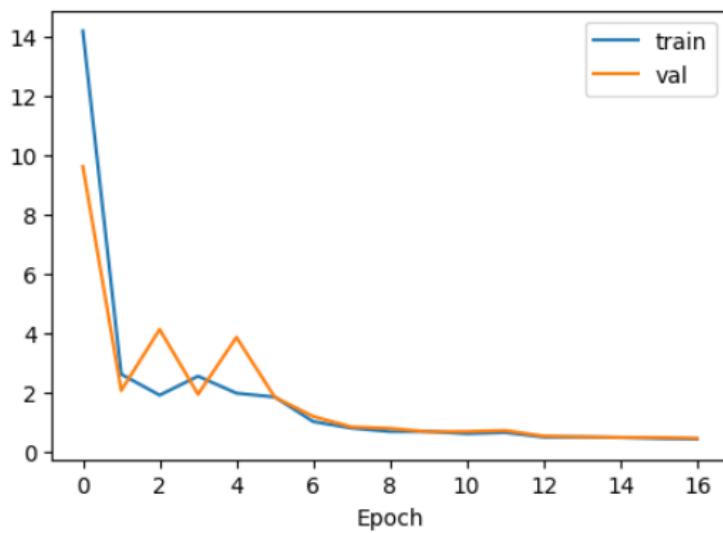
➤ RMSprop - Graph for learning rate 0.01:

❖ Epochs 17:

Accuracy Evolution



Loss Evolution

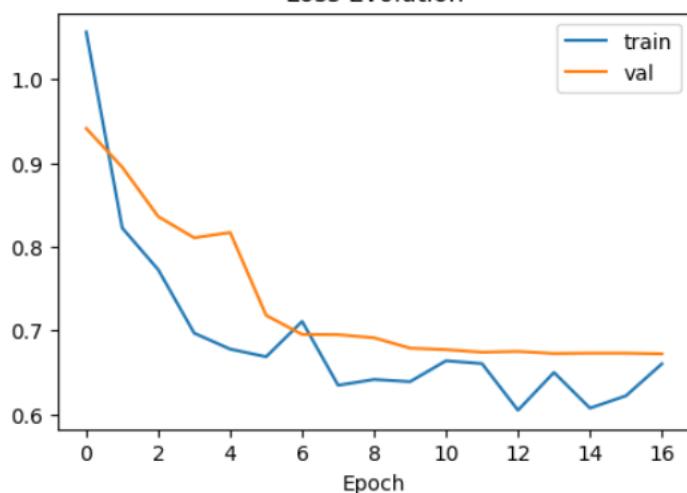


Test accuracy = 90.83%

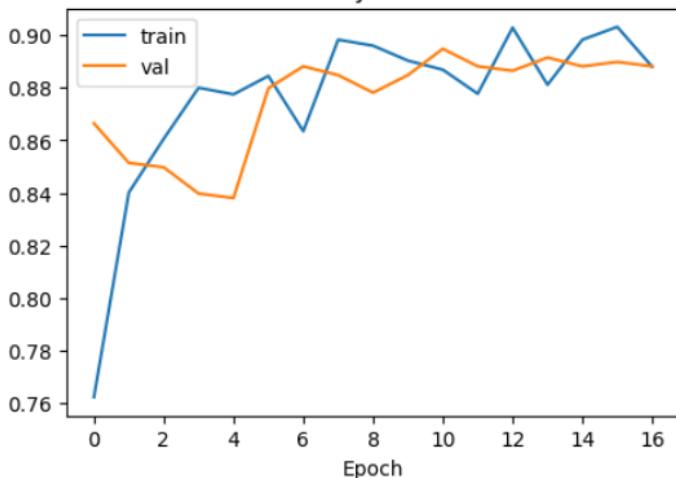
➤ **RMSprop - Graph for learning rate 0.001:**

❖ **Epochs 17:**

Loss Evolution



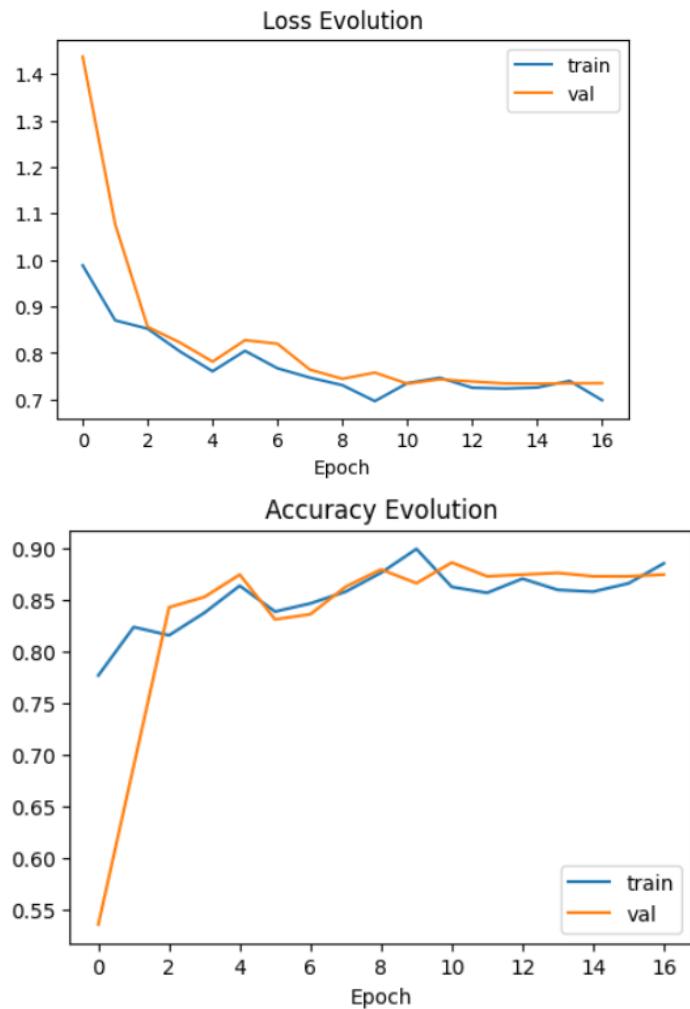
Accuracy Evolution



➤ 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 האופטימיזרים בסעיפים א'-ד', נמצא במודל CNN בסיסי ללא Transfer Learning האלגוריתם עם **Adam** עם **Learning Rate = 0.001** ו-**Epochs = 15** הביא לביצועים הטובים ביותר, עם דיוק אימון של **0.9770** ודיוק אימון של **0.9938**.
זוק הטענות יציבה ומהירה.

לכן, לצורך סעיף זה, הופעל מנגנון EarlyStopping על המודל CNN בסיסי ללא Adam עם Learning patience=5 :

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

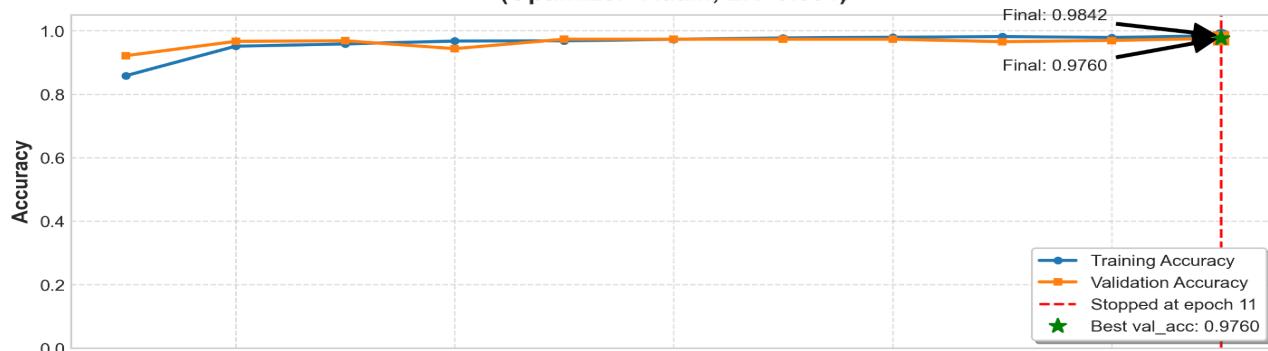
תוצאה:

- האימון נעצר לאחר **11 אפוקים בלבד**
- Accuracy: **0.9760** Validation Accuracy • (Epoch 11 מיטבי : 0.9760 (ב- Epoch 8 מיטבי : 0.0755 Validation Loss •
- חסכו בזמן אימון : כ- 63.3% יחסית להרצת מלאה של 30 אפוקים

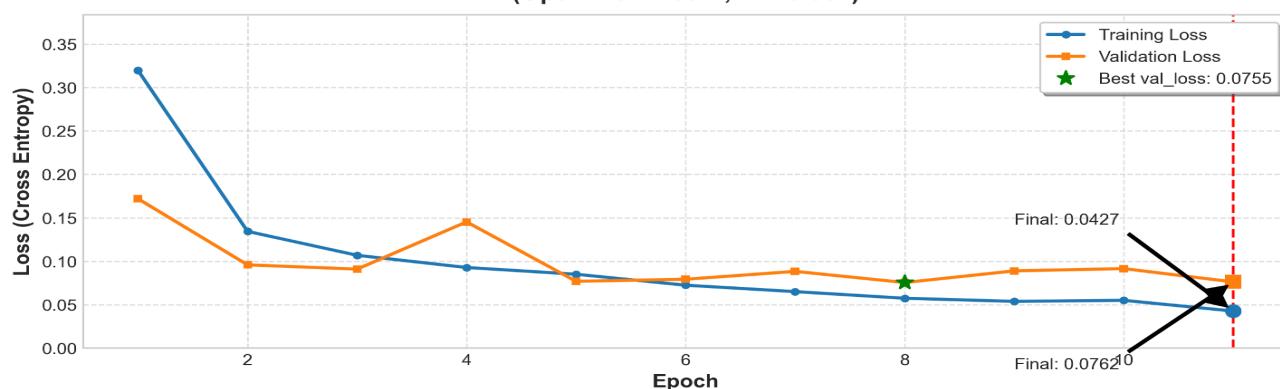
שיפור ביצועים:

למרות ירידת זינחה של כ- 1.4% בדיקת האימונות בהשוואה להרצת ללא עצירה מוקדמת, הושג חסכו ניכר בזמן אימון ומניעת Overfitting – כמובן, התקבל איזון טוב יותר בין דיוק לבין יעילות. לכן, ניתן לקבוע שהשימוש ב- EarlyStopping שיפר את ההשתנות הכלולת של המודל בתנאי אימונו מציאותיים.

Training and Validation Accuracy with Early Stopping
(Optimizer=Adam, LR=0.001)



Training and Validation Loss with Early Stopping
(Optimizer=Adam, LR=0.001)

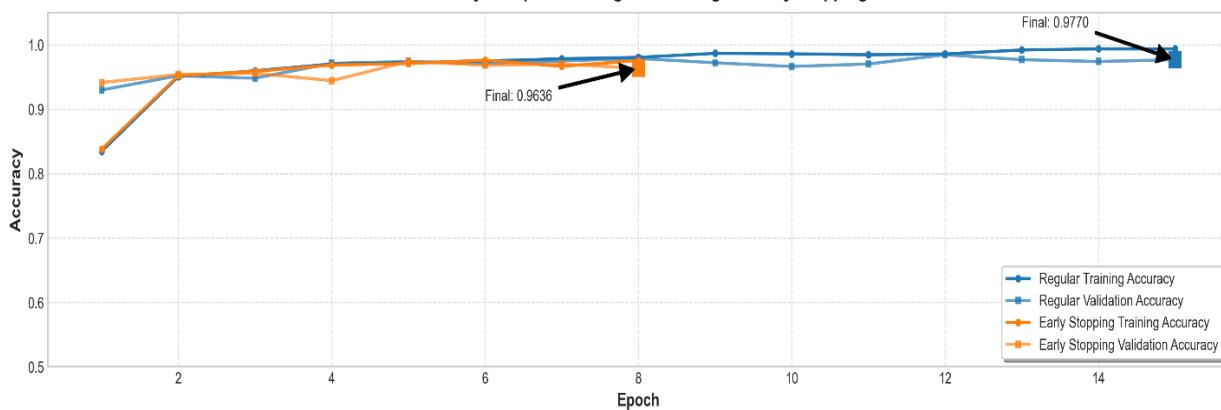


Early Stopping Summary:

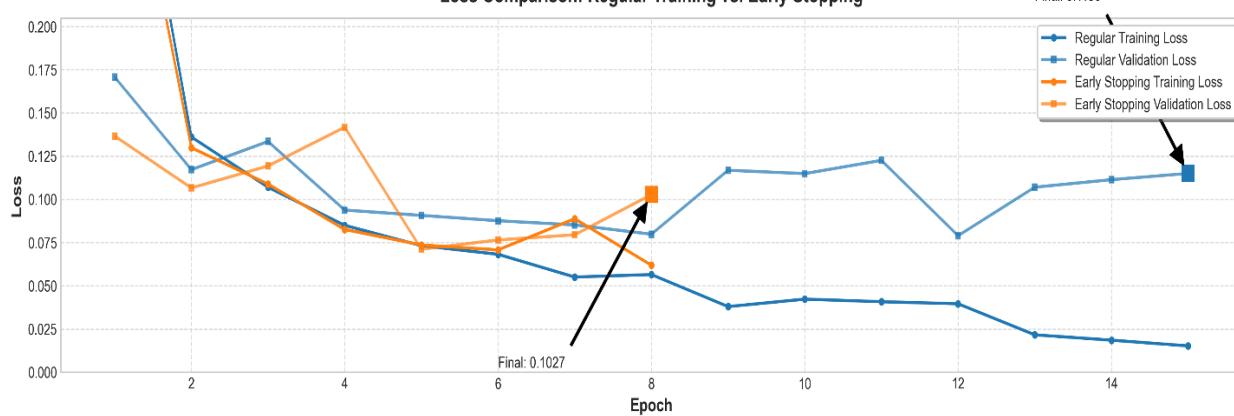
- Stopped at epoch 11 (requested max: 30)
 - Final accuracy: 0.9760 (train: 0.9842)
- Best validation accuracy: 0.9760 at epoch 11
 - Best validation loss: 0.0755 at epoch 8
- Total Images: 1043 training, 1043 validation
 - Training time savings: 63.3%

Early Stopping Impact Analysis for Adam (LR=0.001)

Accuracy Comparison: Regular Training vs. Early Stopping



Loss Comparison: Regular Training vs. Early Stopping



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

משימה 4

שימוש באמצעות שינוי הרשות ללא TRANSFER LEARNING (משימה 1) פתרון לסיווג 3 קטגוריות: (1) אין דלקת (2) יש דלקת חיידקית (3) יש דלקת נגיפית. שימוש לב שיש להשתמש במרקבה זה-ב-softmax במצב הרשות. חזרו על משימה 3 עבור הרשות במשימה זאת, והציגו עבור הפתרון הטוב ביותר שהשגתם את ה-CONFUSION MATRIX על סט הבדיקה.

במשימה זו נדרשנו להתאים את רשות ה- CNN מהמשימה הראשונה (לא 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.

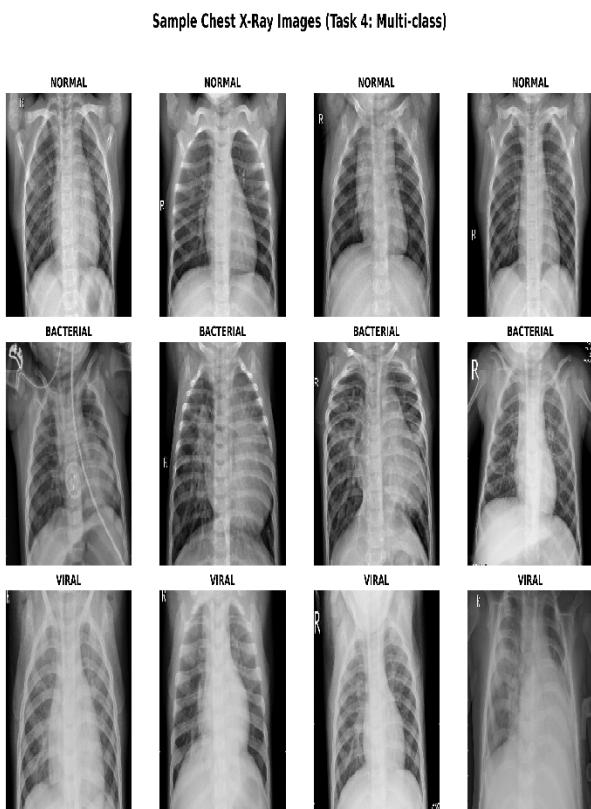
אימון הרשות עם אופטימיזרים שונים

בzdomaה למשימה 3, בוצעו 36 ריצות אימון של המודל – עברו כל שילוב של :

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

הקוד יצר 36 גրפים המציגים את תהליך האימון, כולל :

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

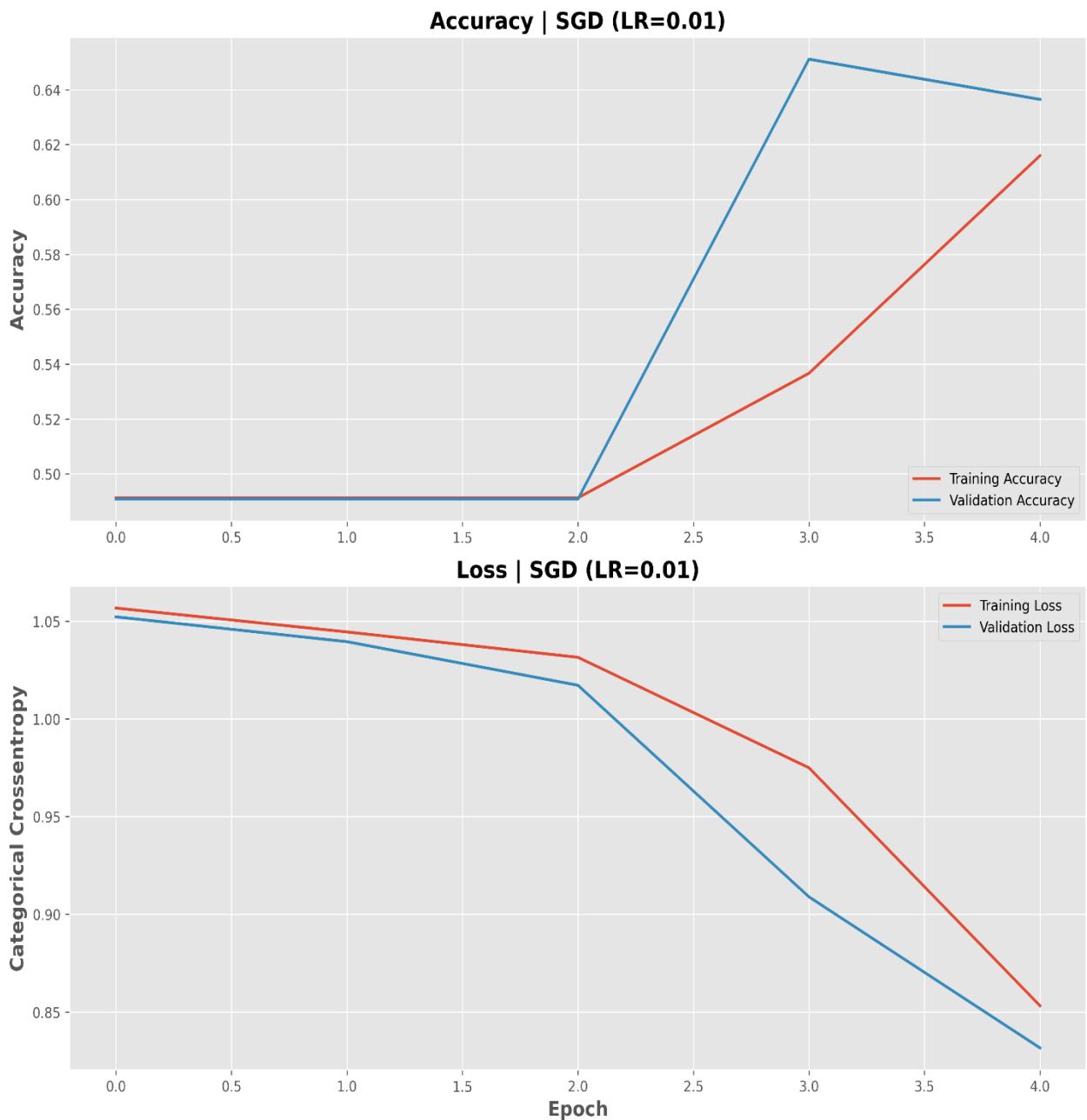


א. אלגוריתם SGD

בדקנו את השפעת SGD הקלasicי ללא Momentum.

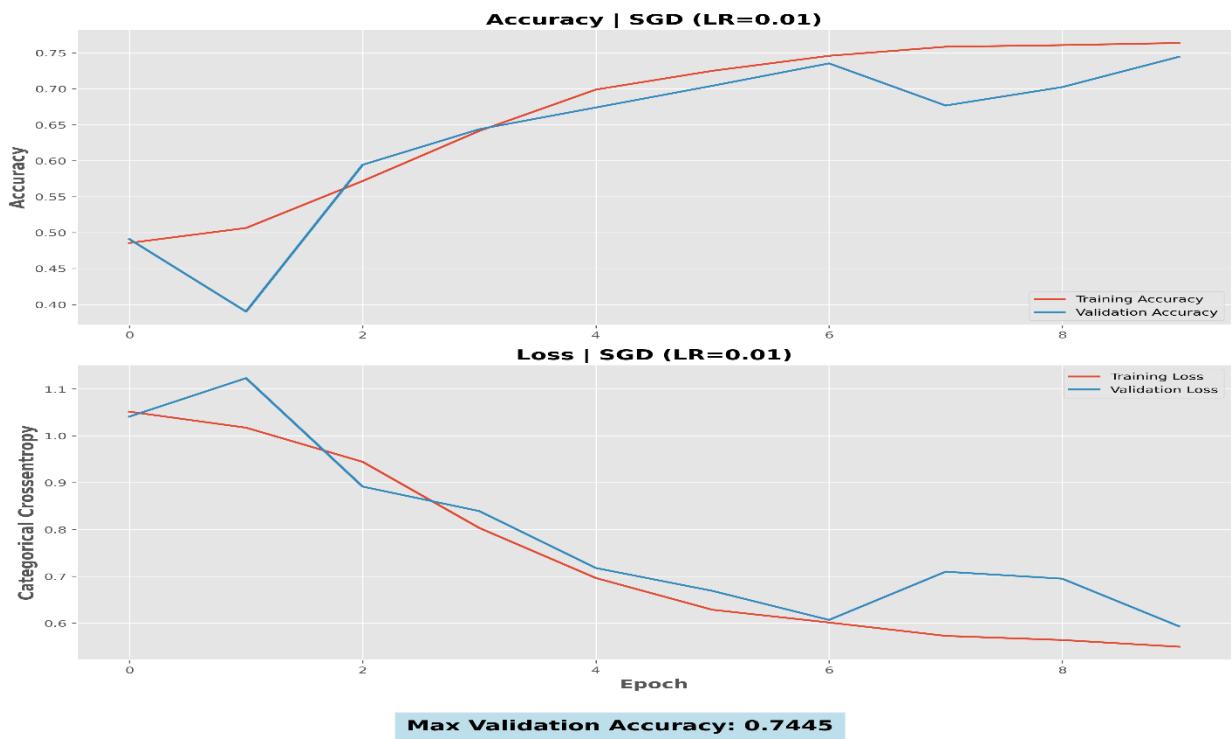
➤ SGD - Graph for learning rate 0.01:

❖ Epochs 5:

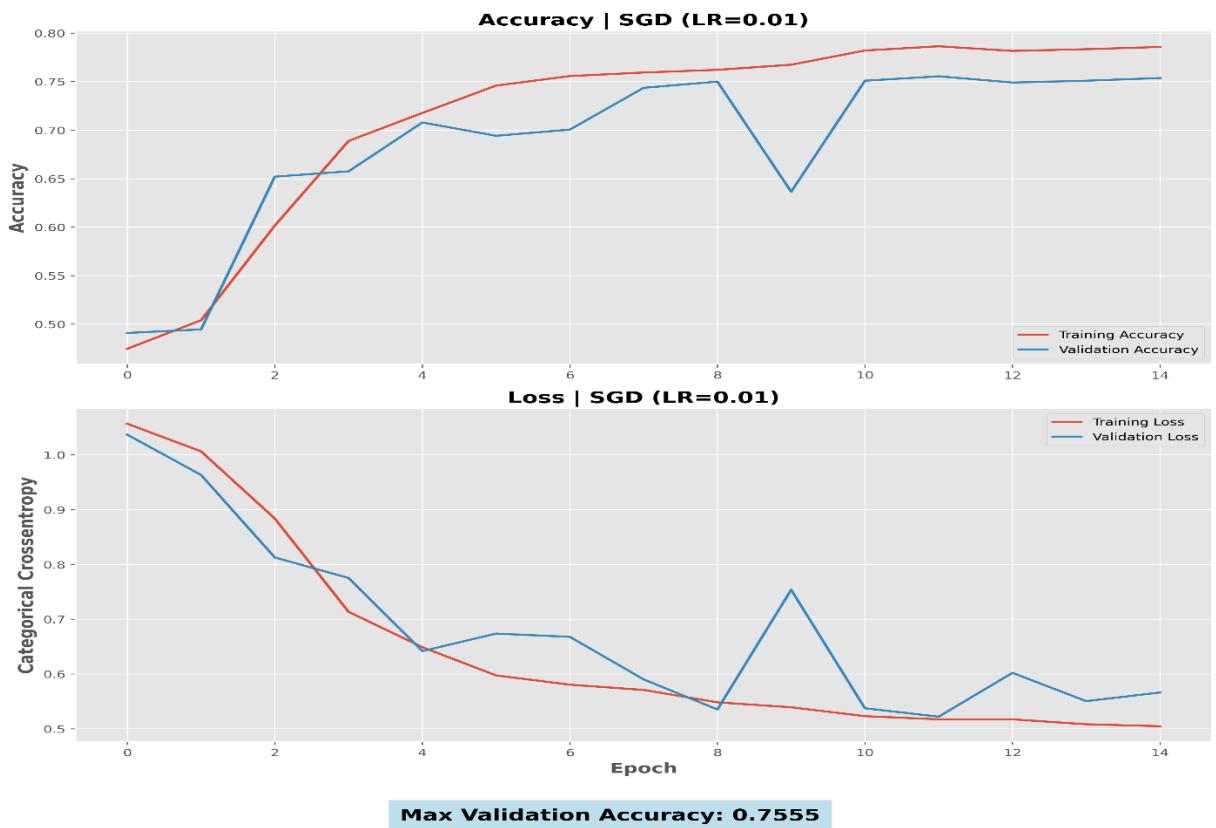


Max Validation Accuracy: 0.6511

❖ Epochs 10:

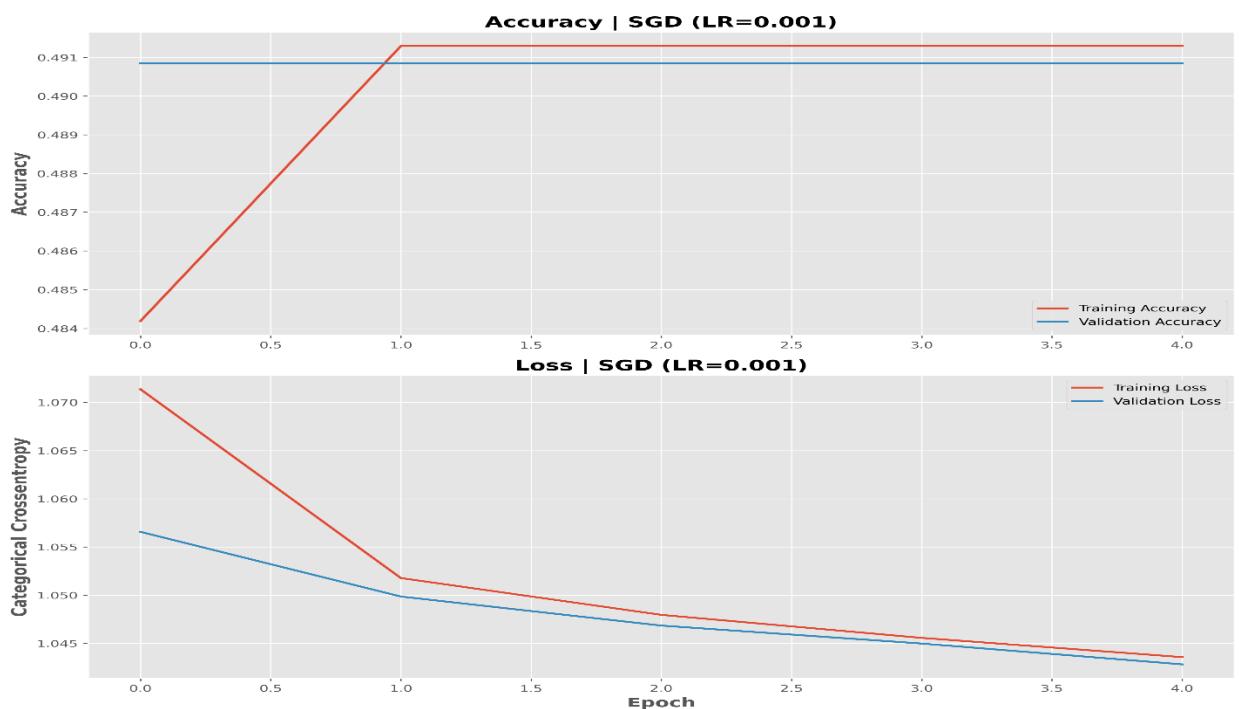


❖ Epochs 15:

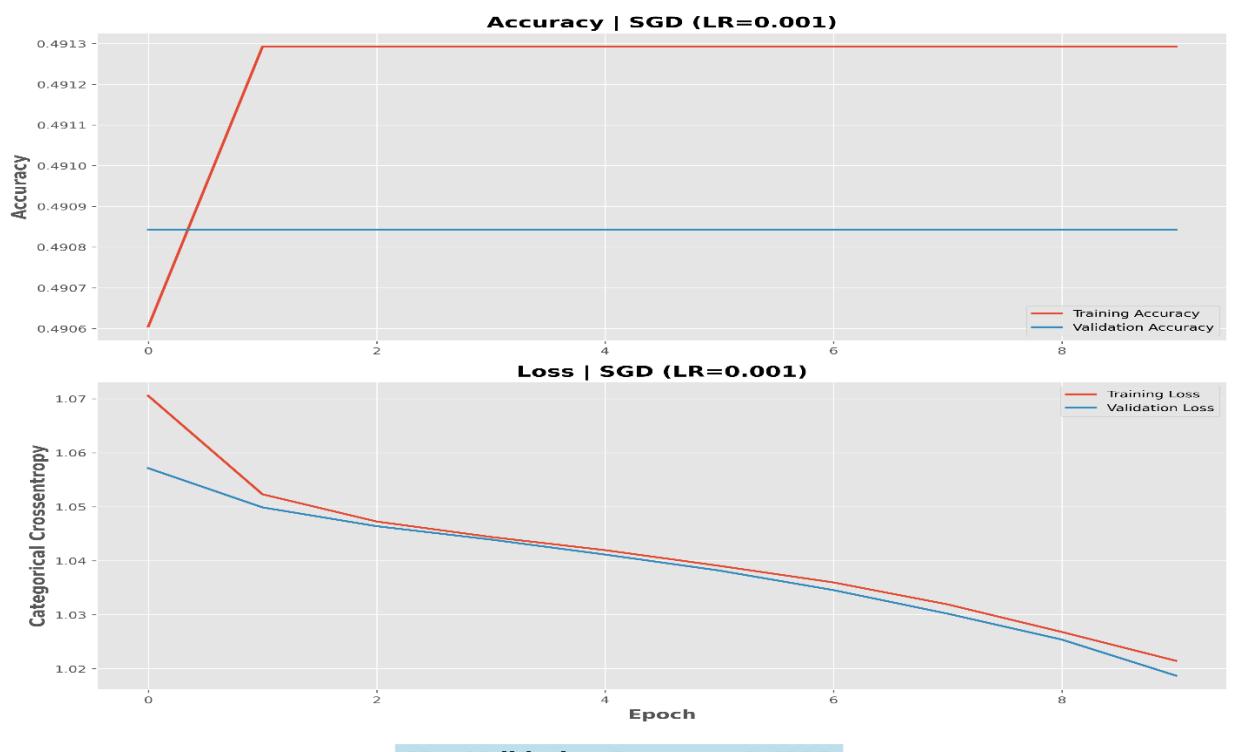


➤ SGD - Graph for learning rate 0.001:

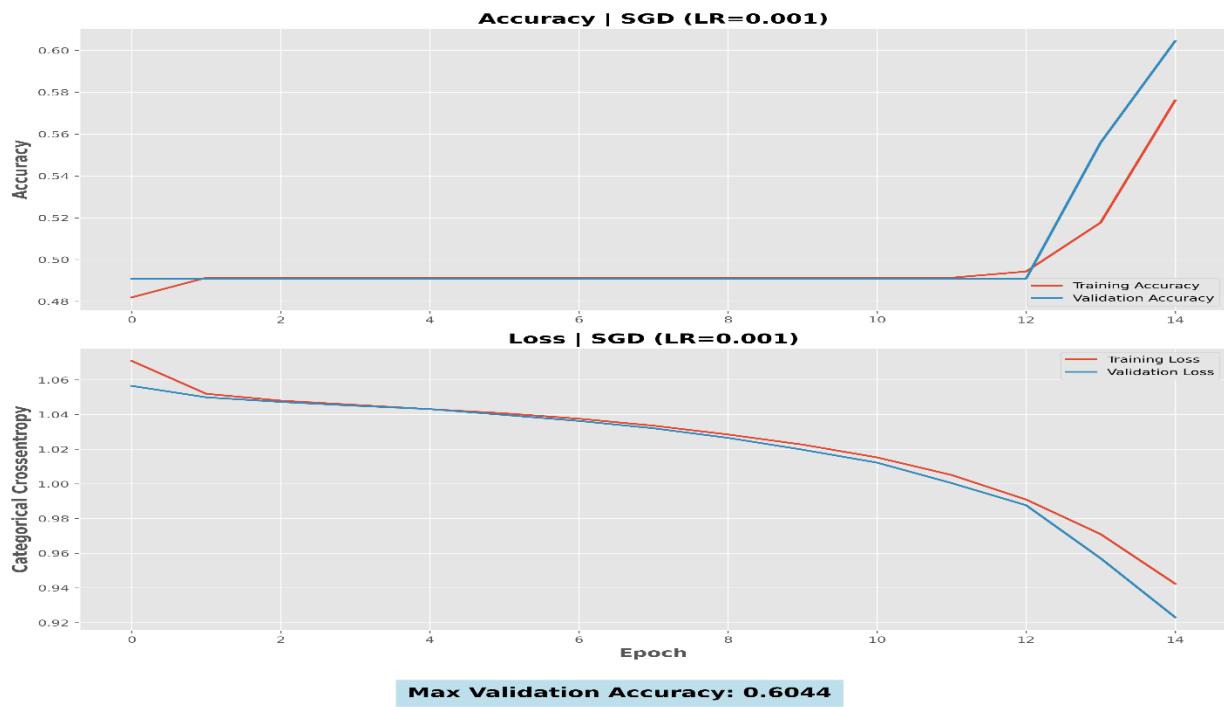
❖ Epochs 5:



❖ Epochs 10:

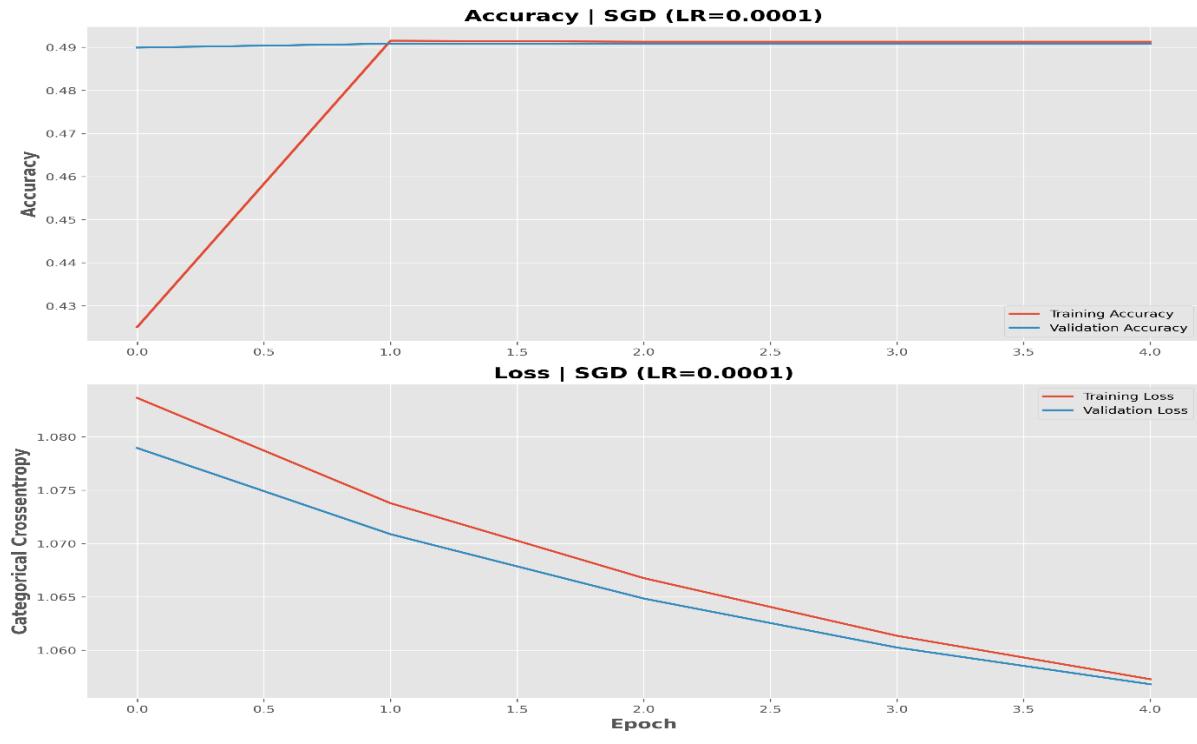


❖ Epochs 15:

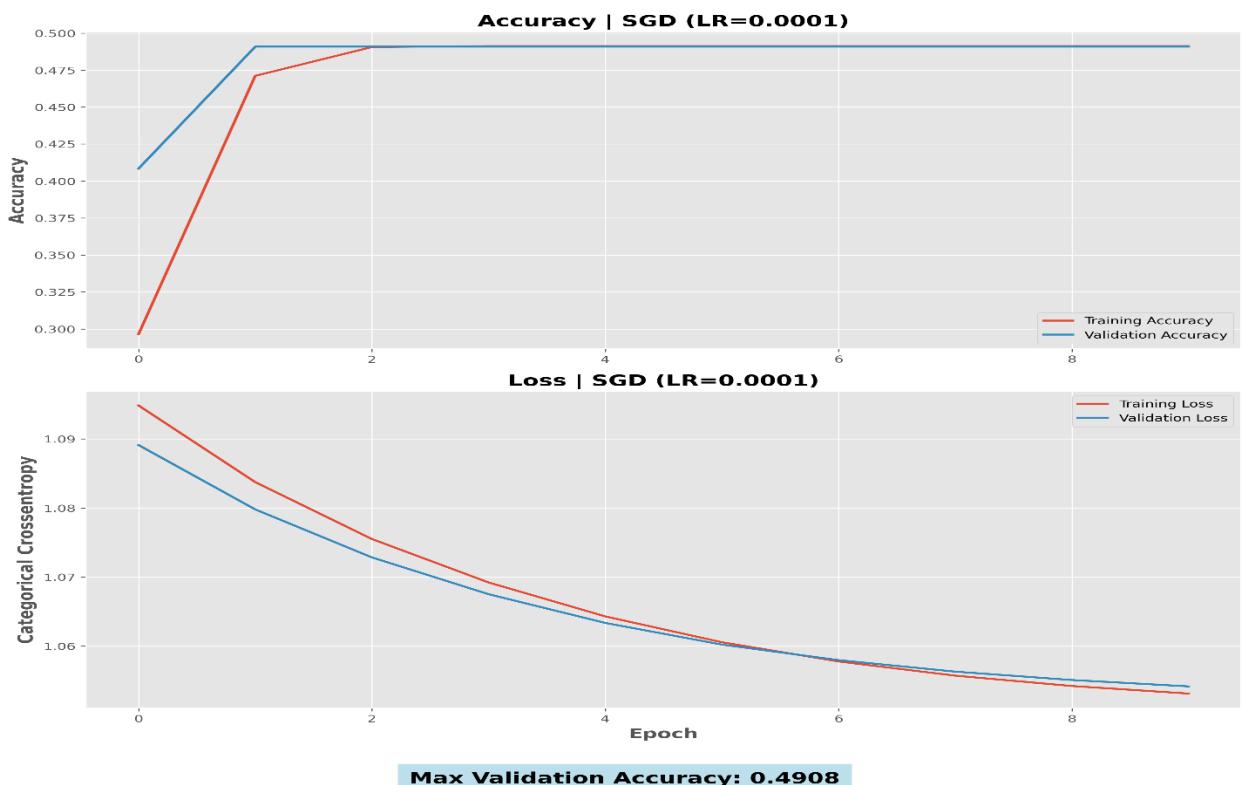


➤ SGD - Graph for learning rate 0.0001:

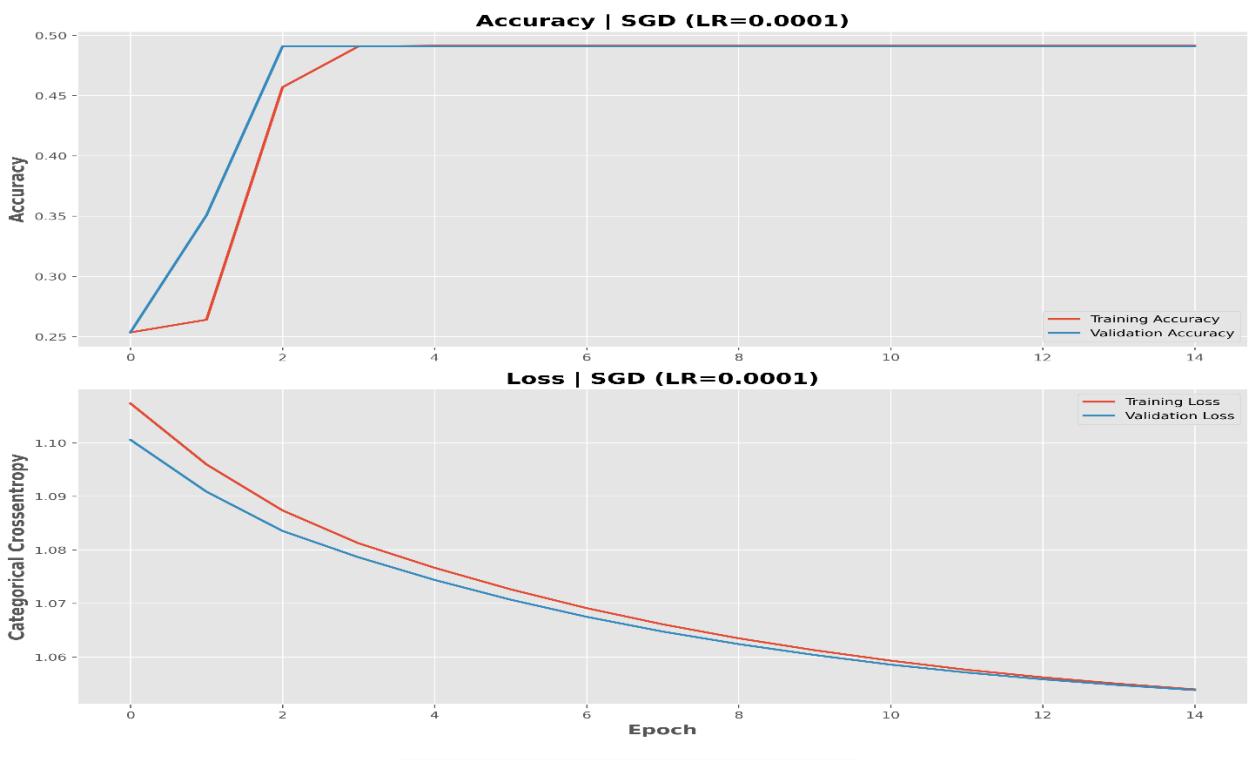
❖ Epochs 5:



❖ Epochs 10:



❖ Epochs 15:

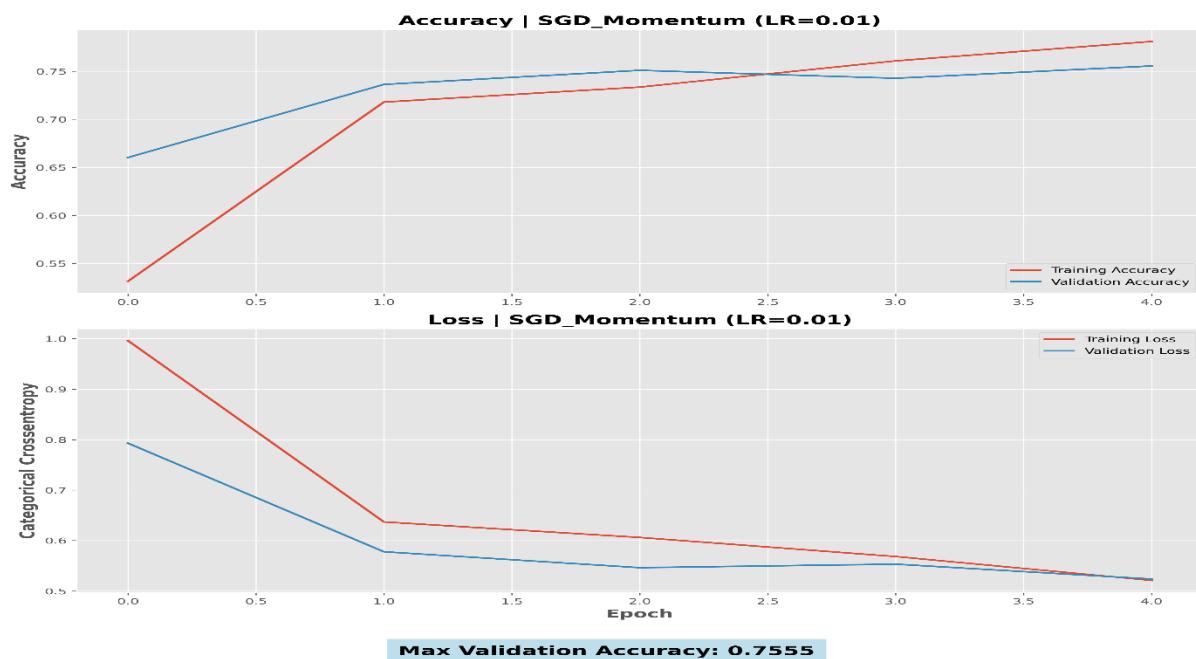


ב. אלגוריתם SGD עם Momentum

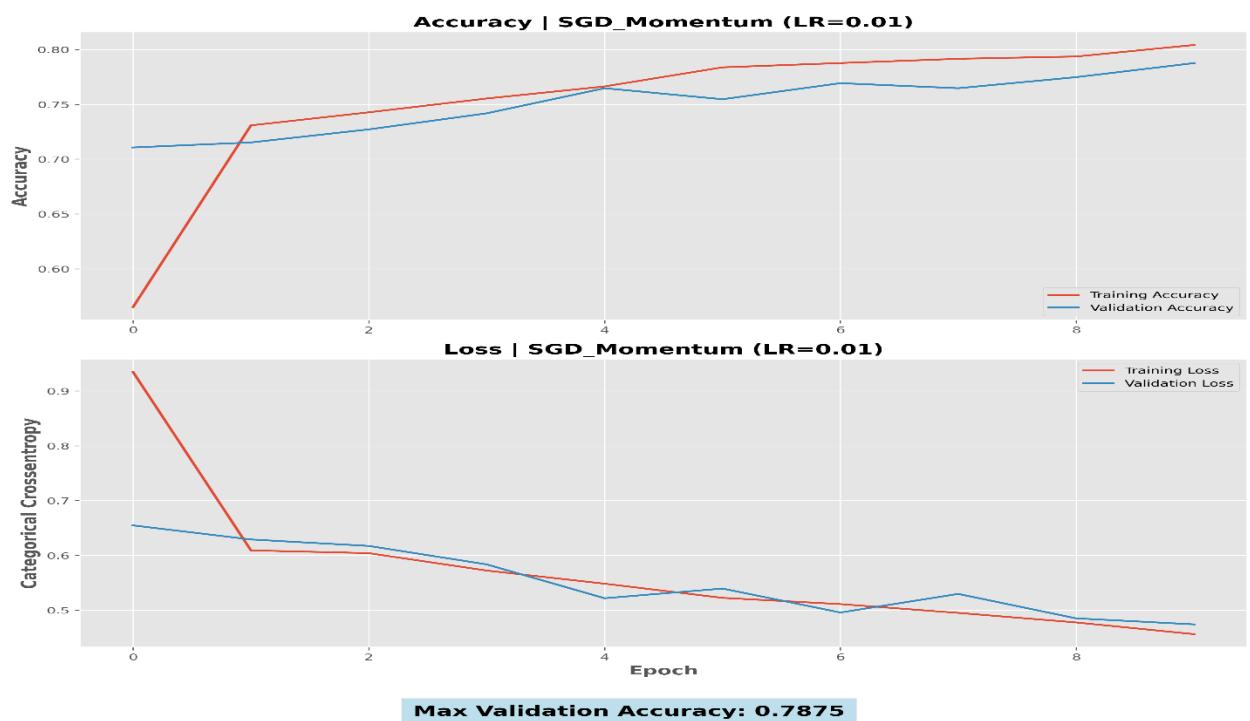
Momentum = 0.9

➤ SGD with Momentum - Graph for learning rate 0.01:

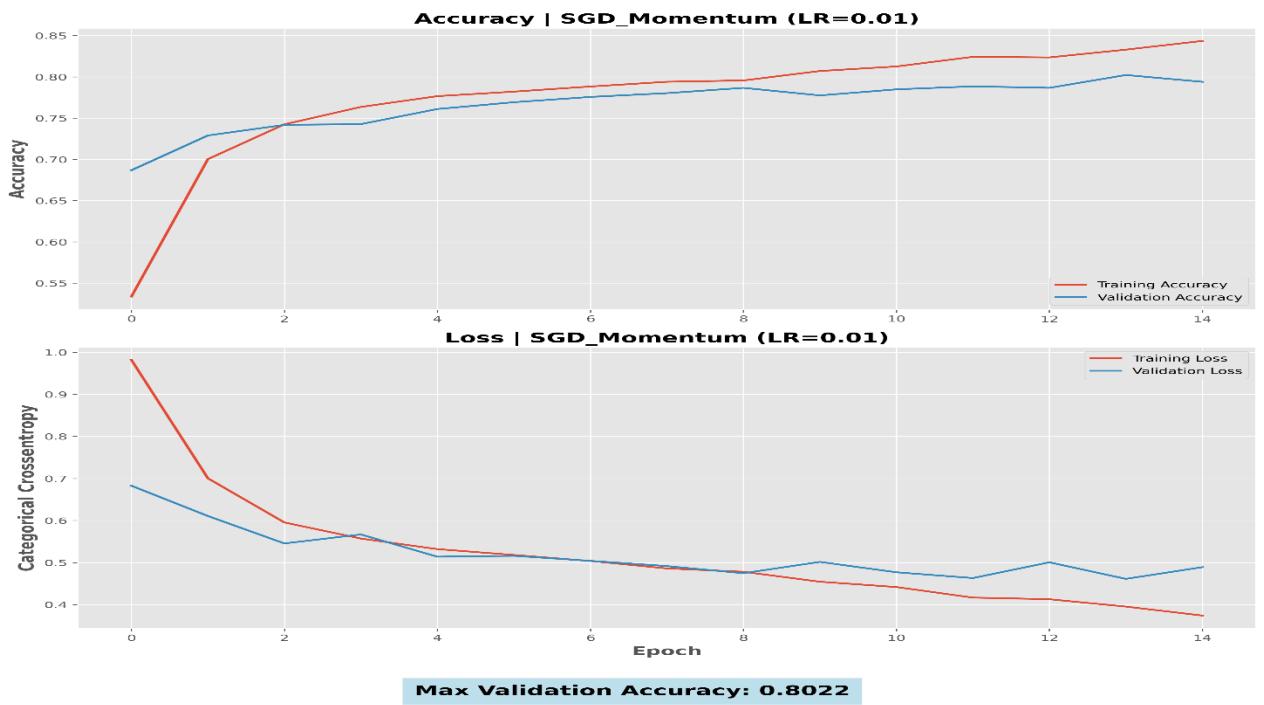
❖ Epochs 5:



❖ Epochs 10:

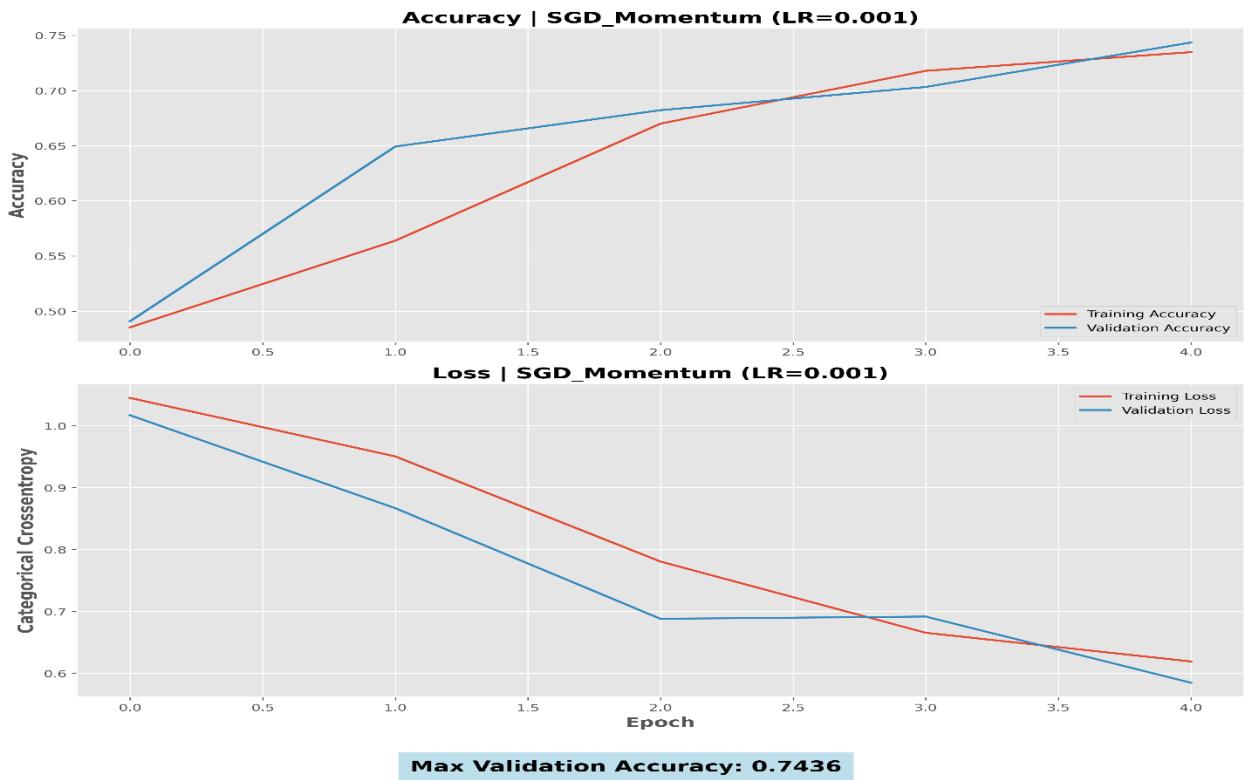


❖ Epochs 15:

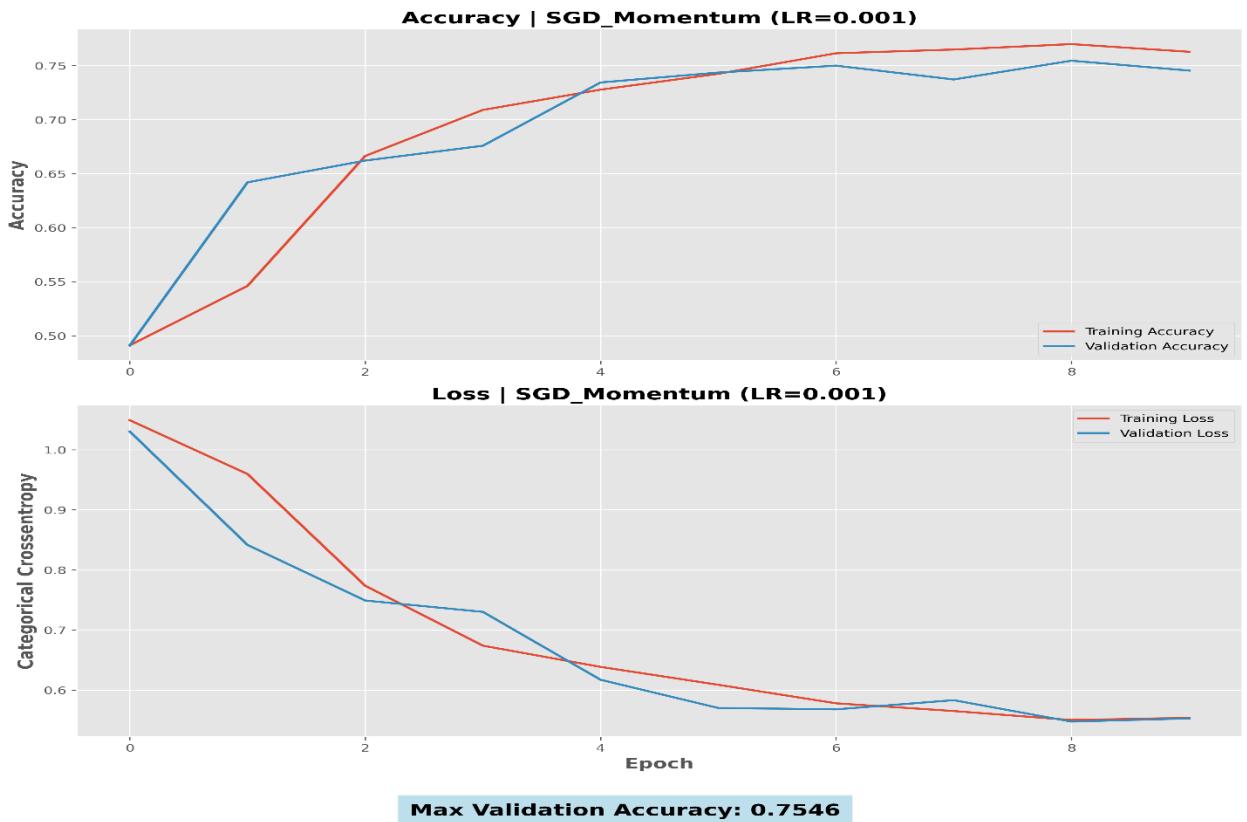


➤ SGD with Momentum - Graph for learning rate 0.001:

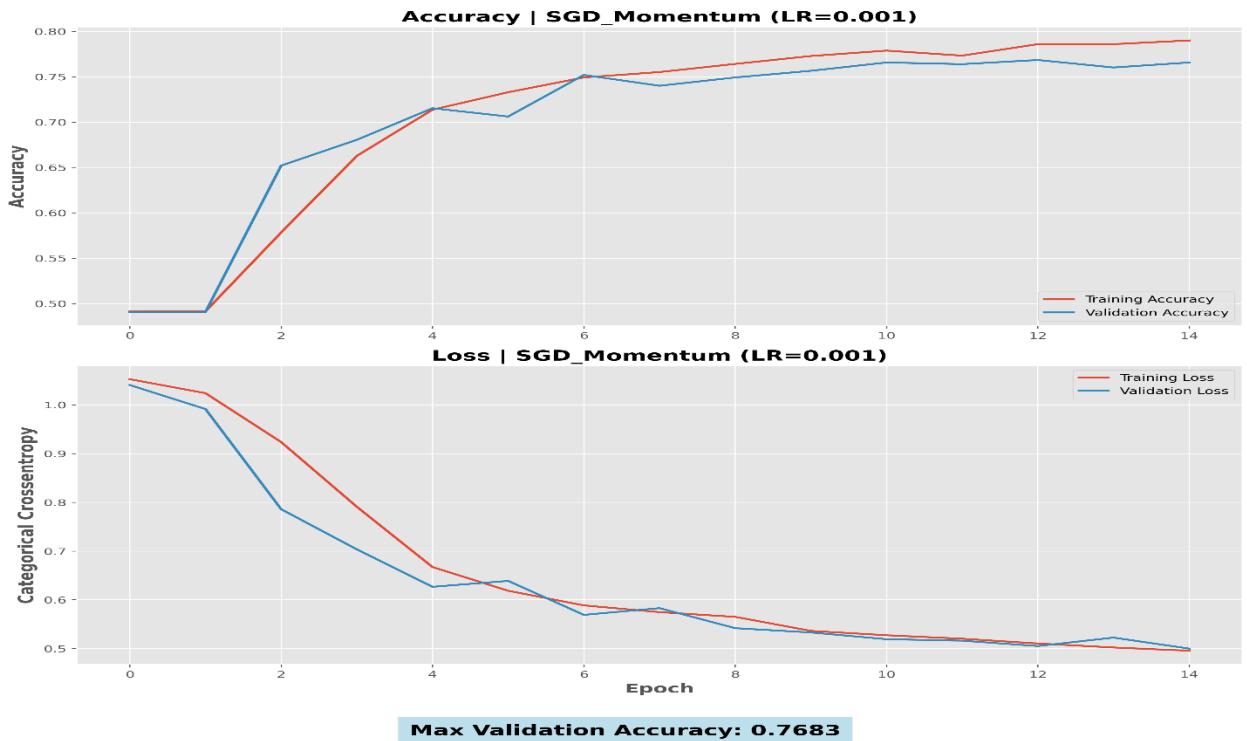
❖ Epochs 5:



❖ Epochs 10:

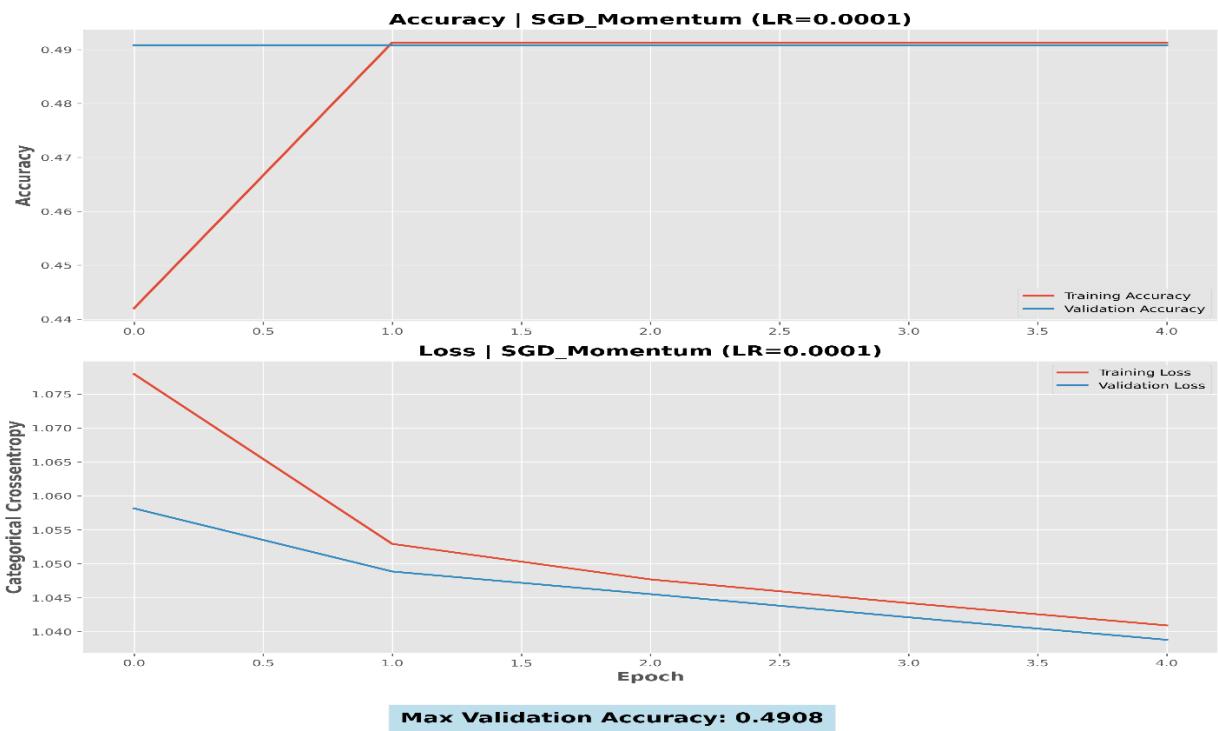


❖ Epochs 15:

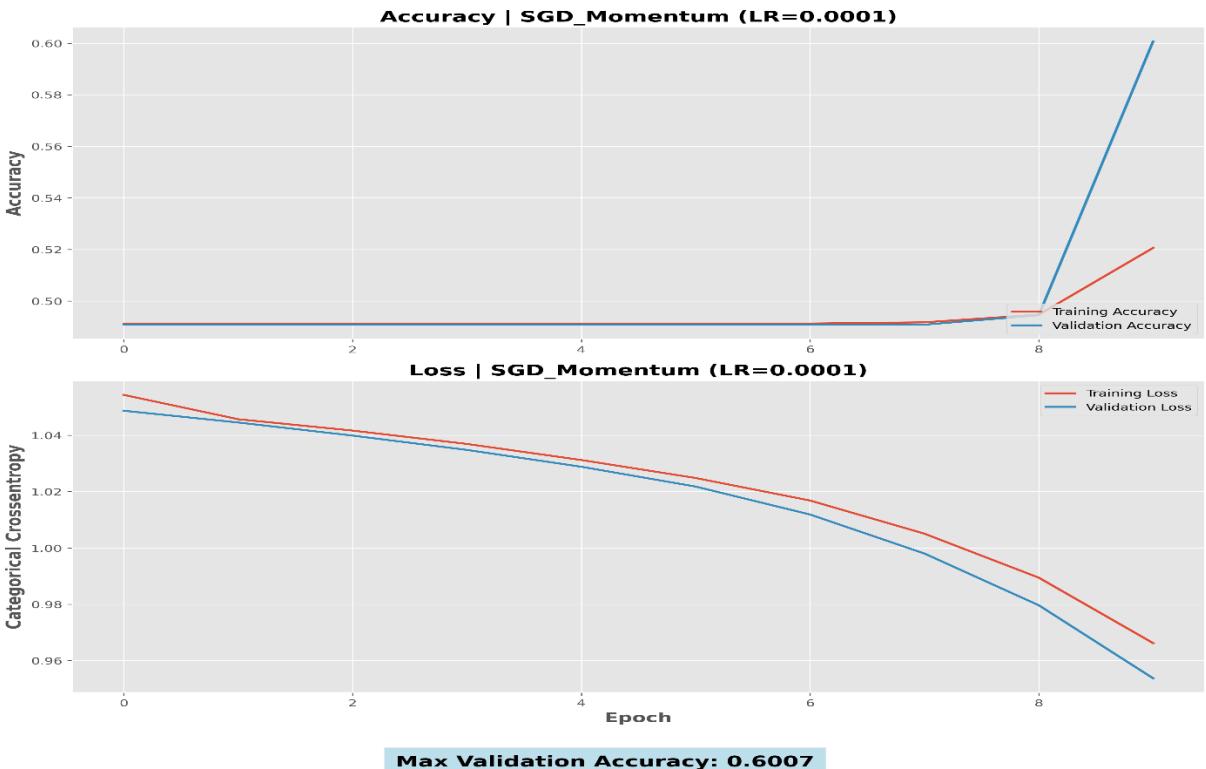


➤ SGD with Momentum - Graph for learning rate 0.0001:

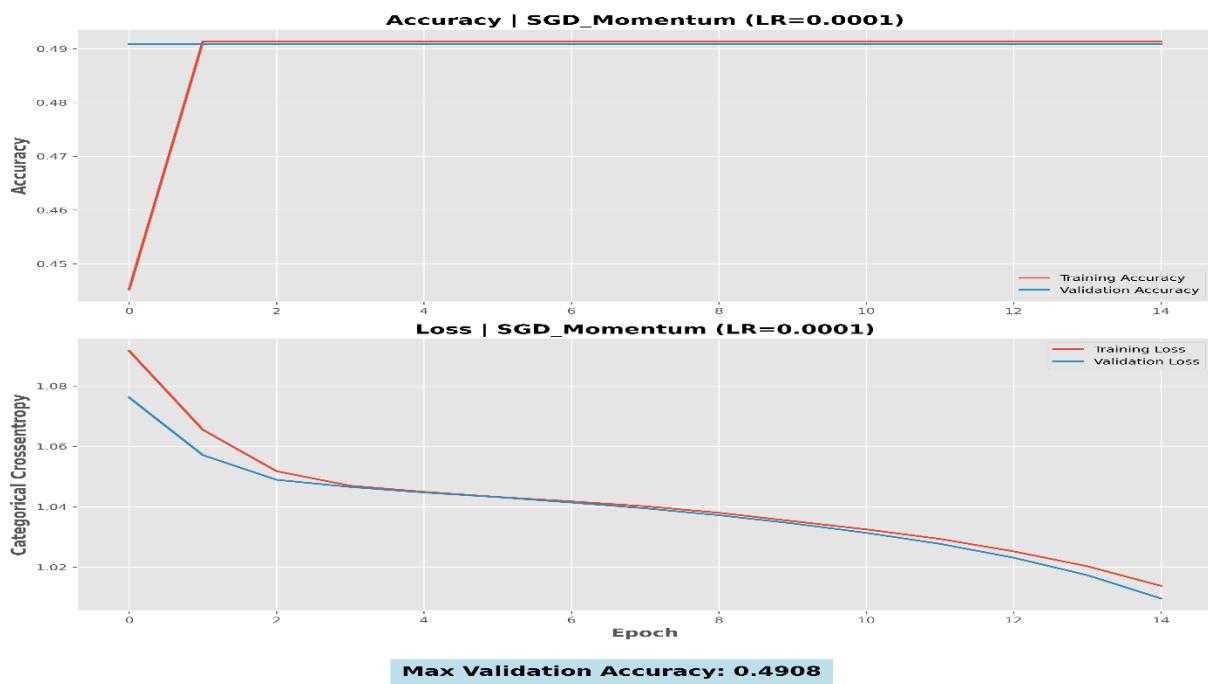
❖ Epochs 5:



❖ Epochs 10:



❖ Epochs 15:

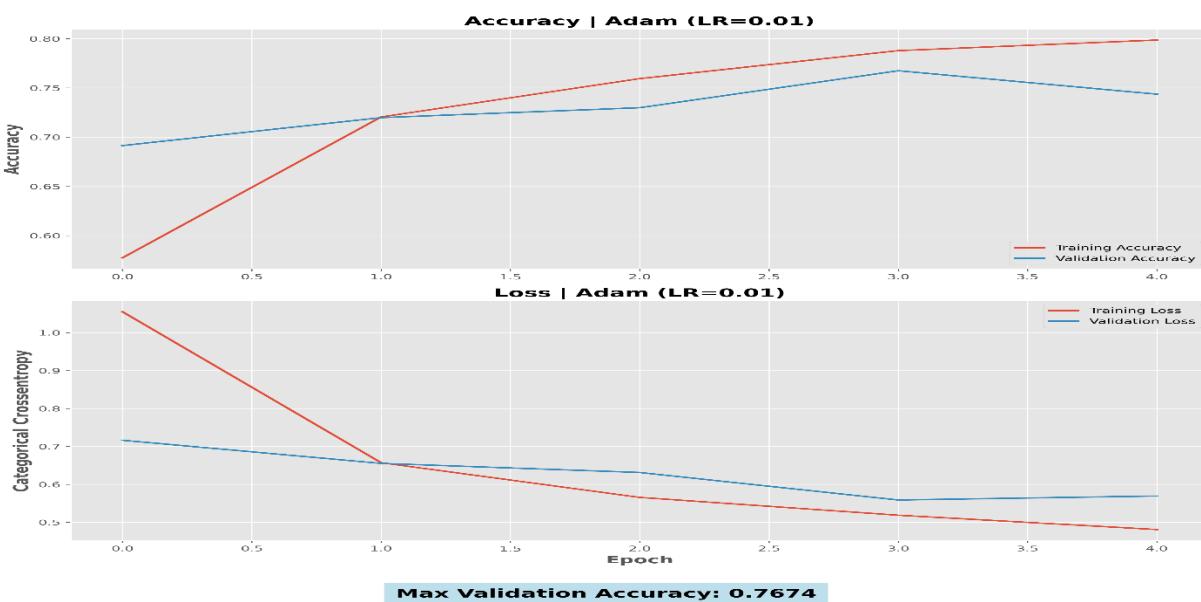


ג. אלגוריתם Adam

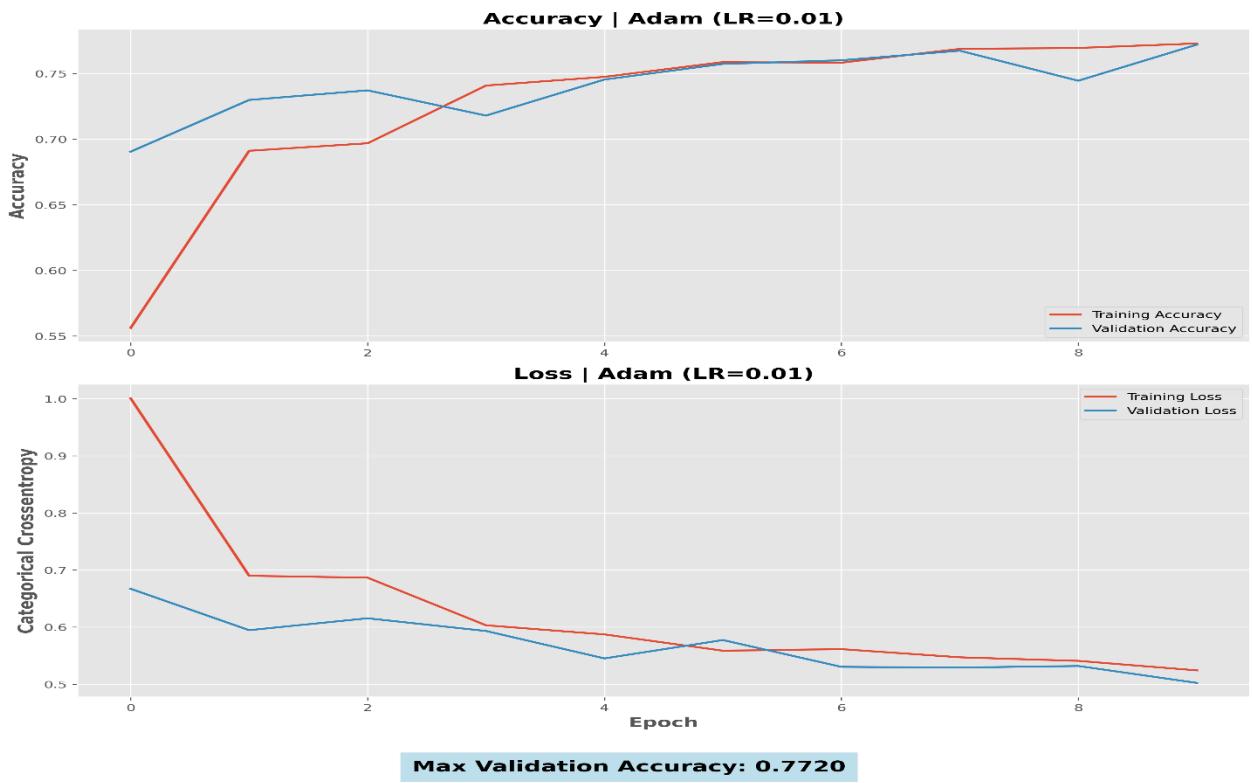
- Adaptive Learning Rate הופולריים למיניהם – משלב **Momentum**

➤ Adam - Graph for learning rate 0.01:

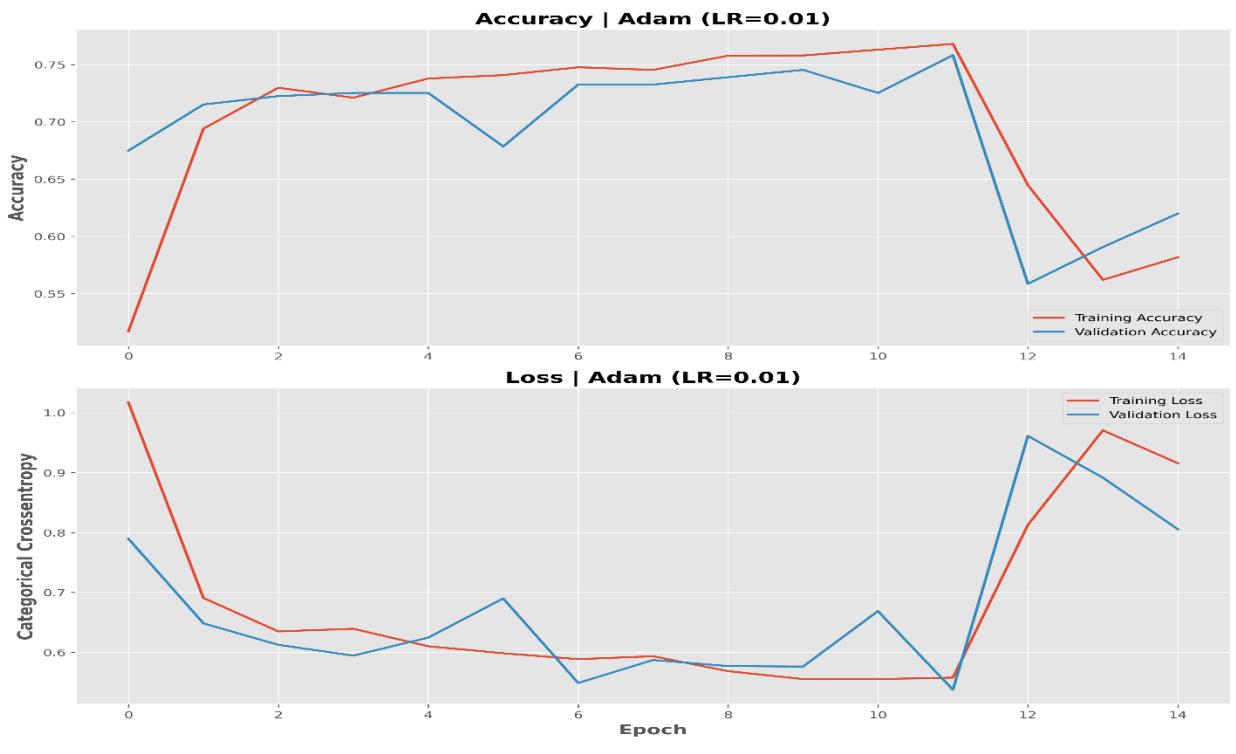
❖ Epochs 5:



❖ Epochs 10:

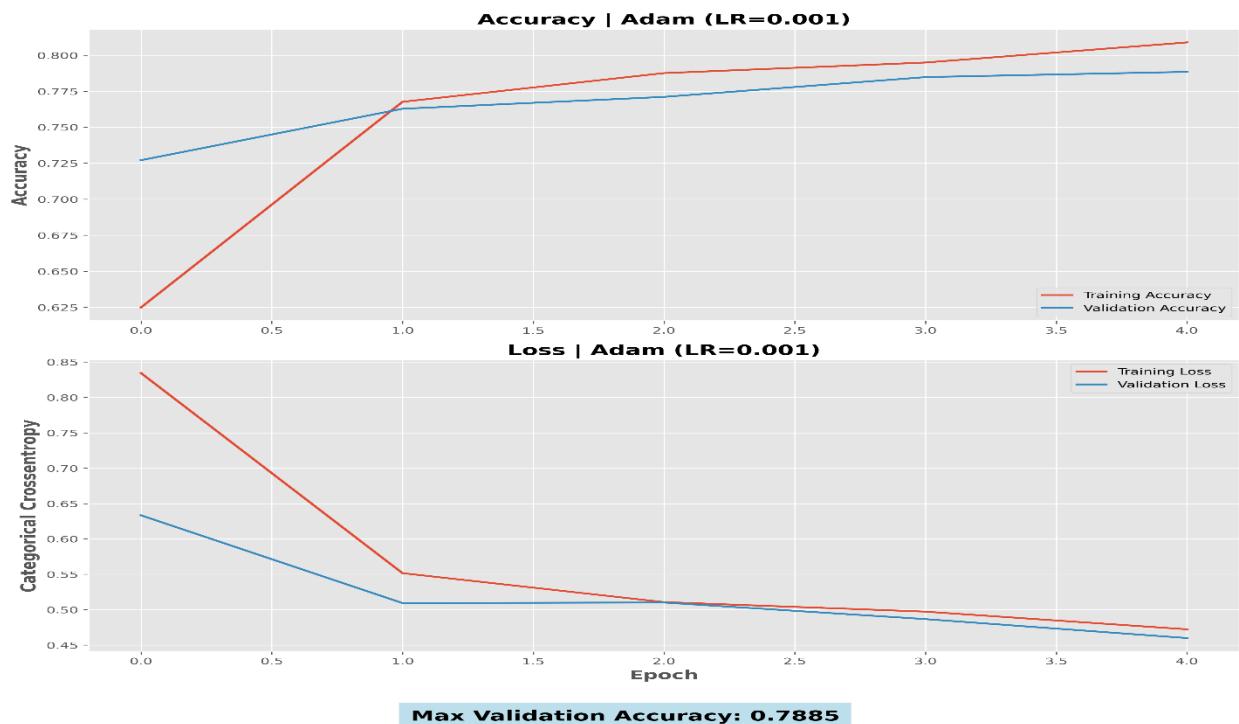


❖ Epochs 15:

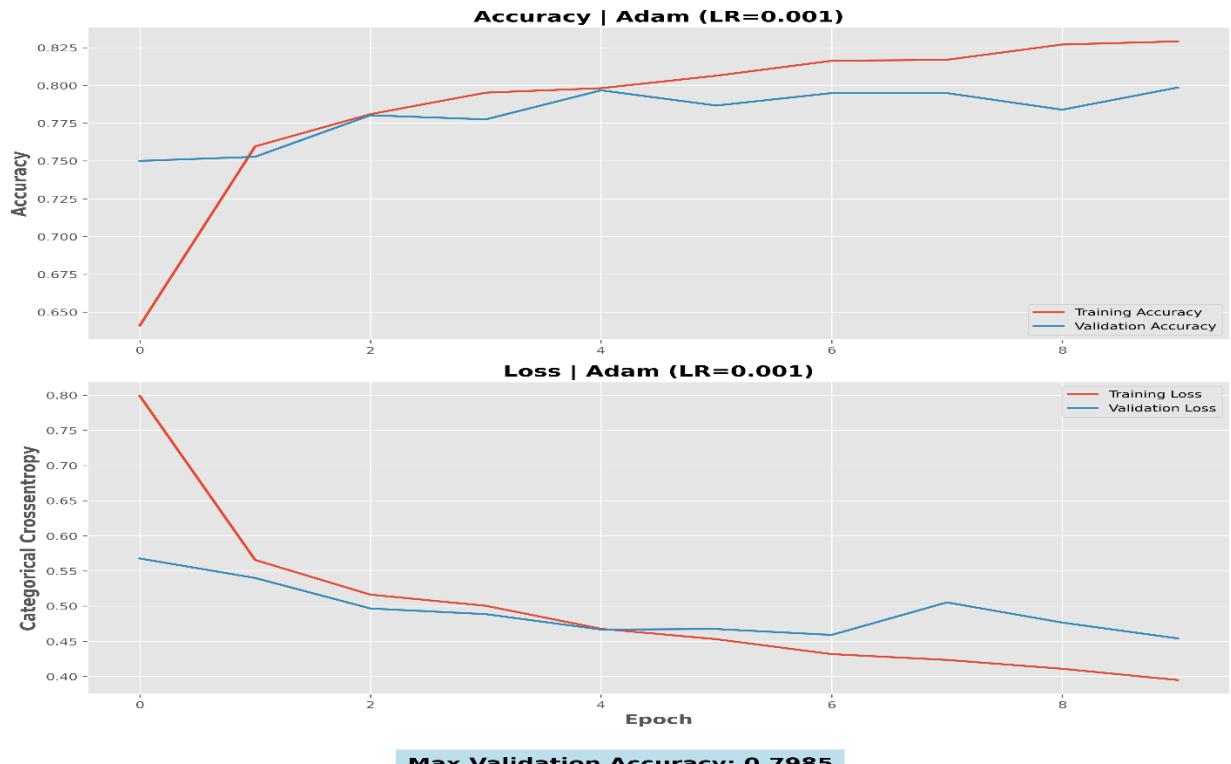


➤ Adam - Graph for learning rate 0.001:

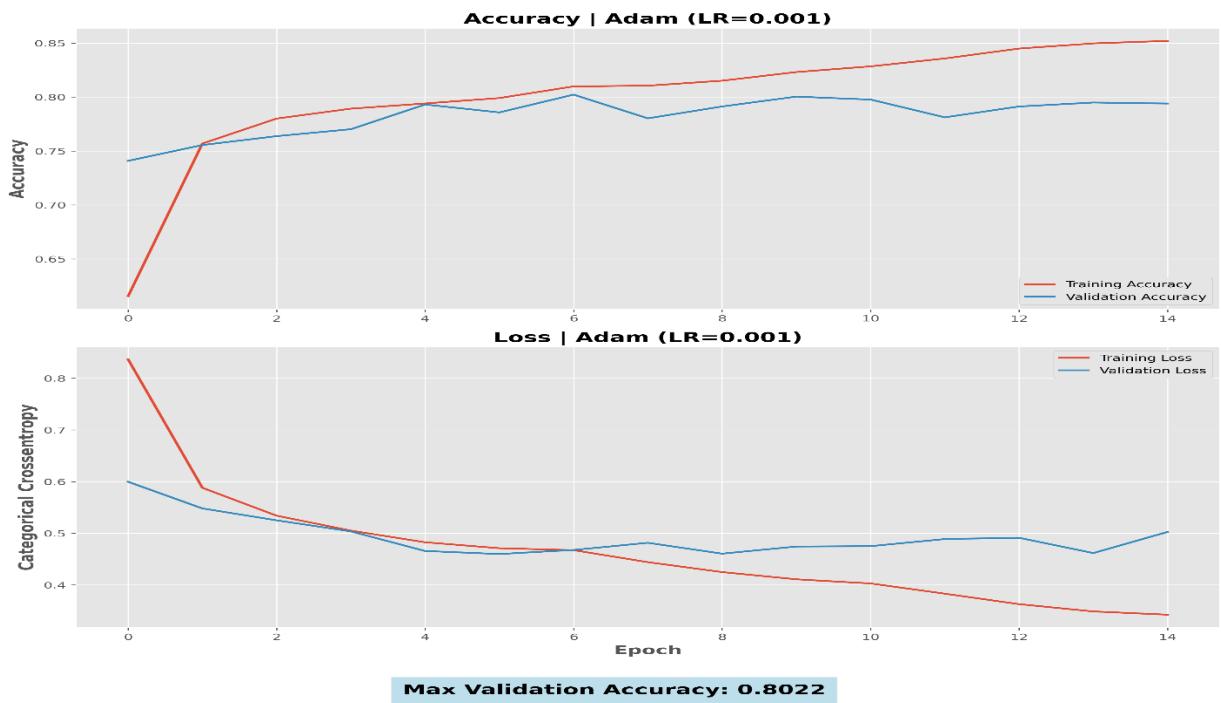
❖ Epochs 5:



❖ Epochs 10:

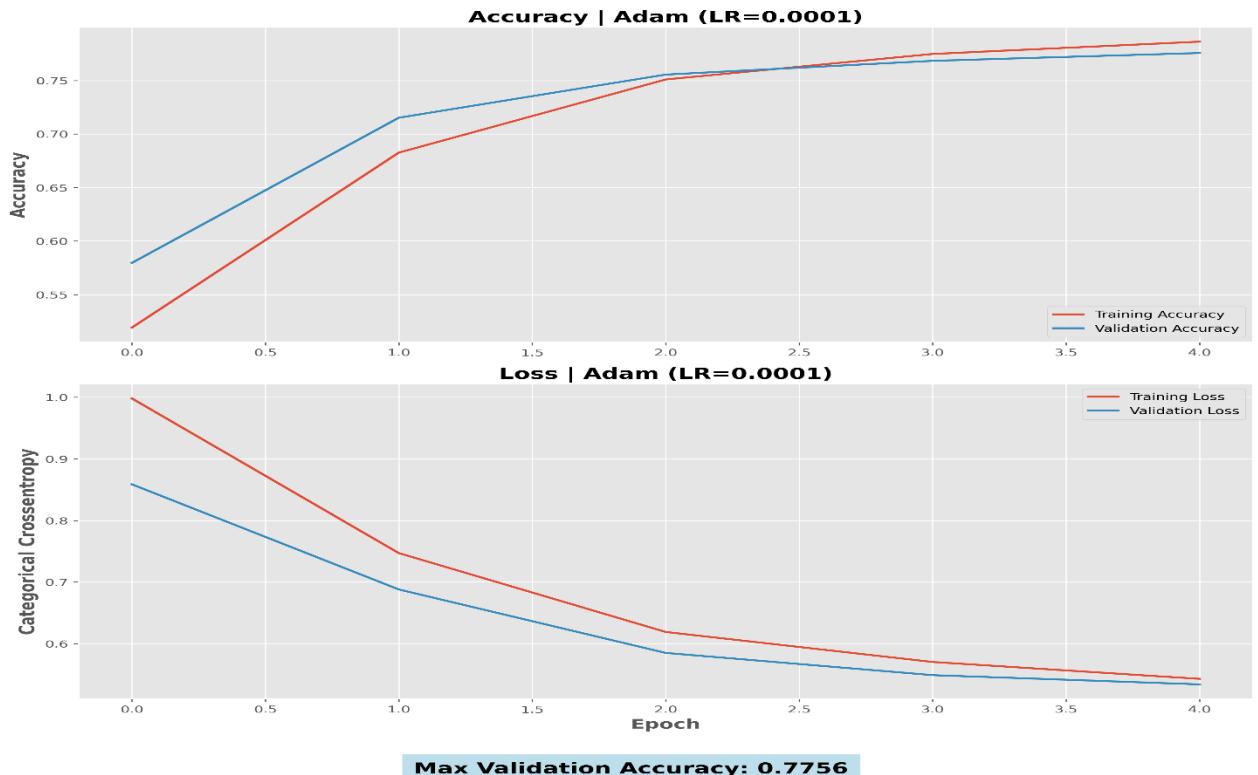


❖ Epochs 15:

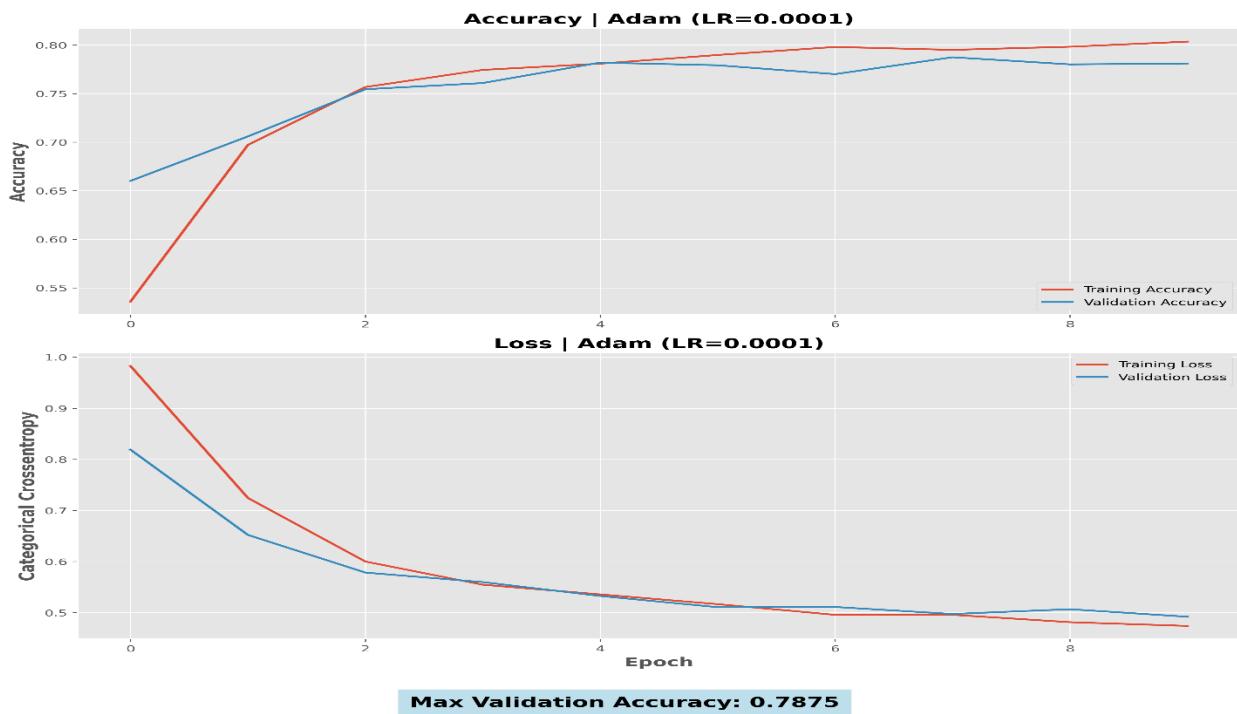


➤ Adam - Graph for learning rate 0.0001:

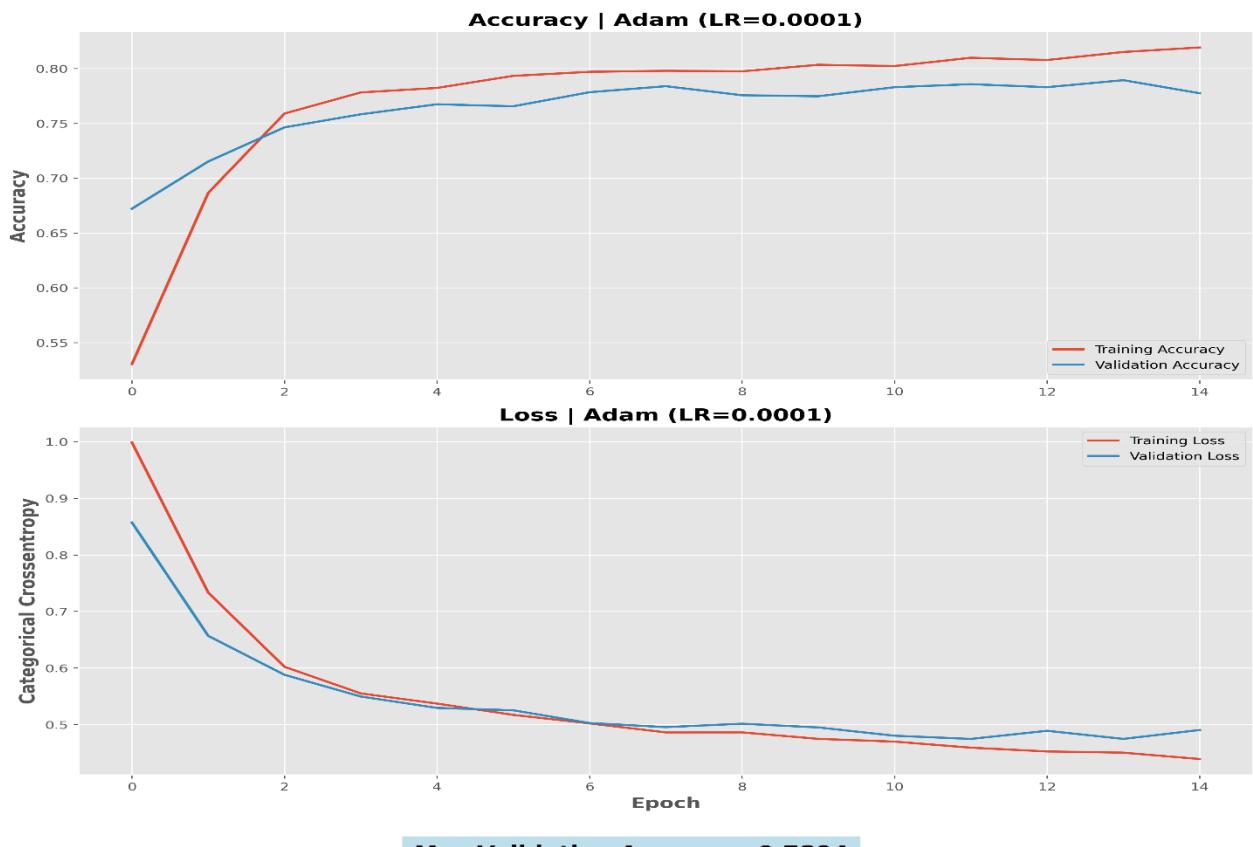
❖ Epochs 5:



❖ Epochs 10:



❖ Epochs 15:

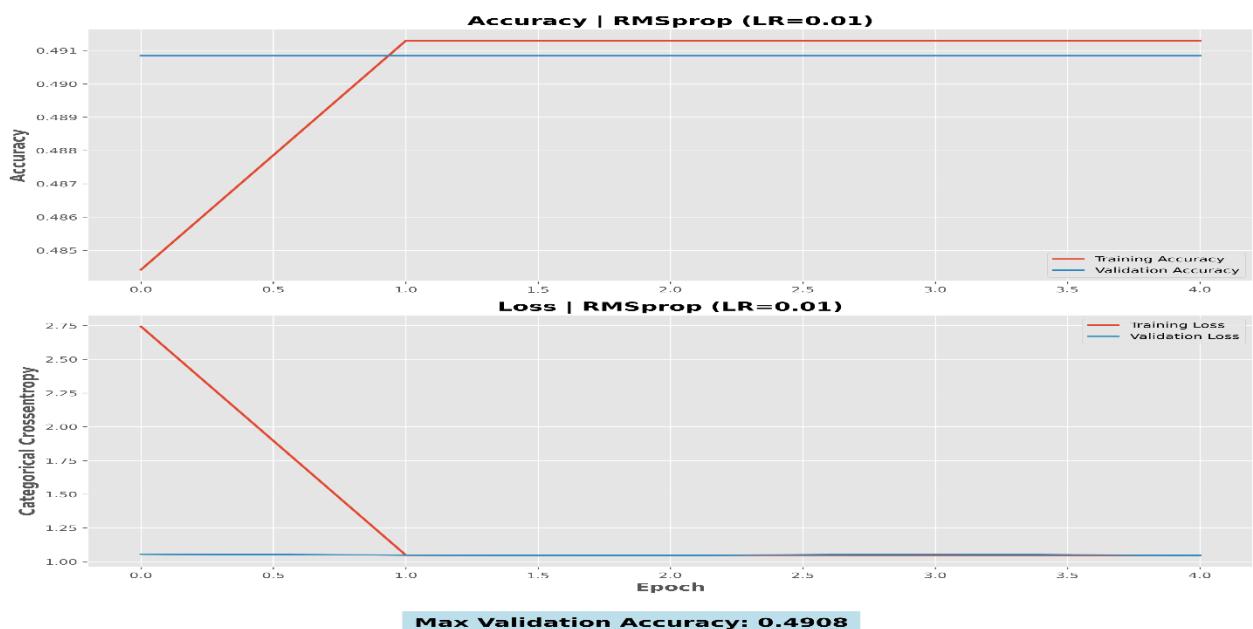


ד. אלגוריתם RMSprop

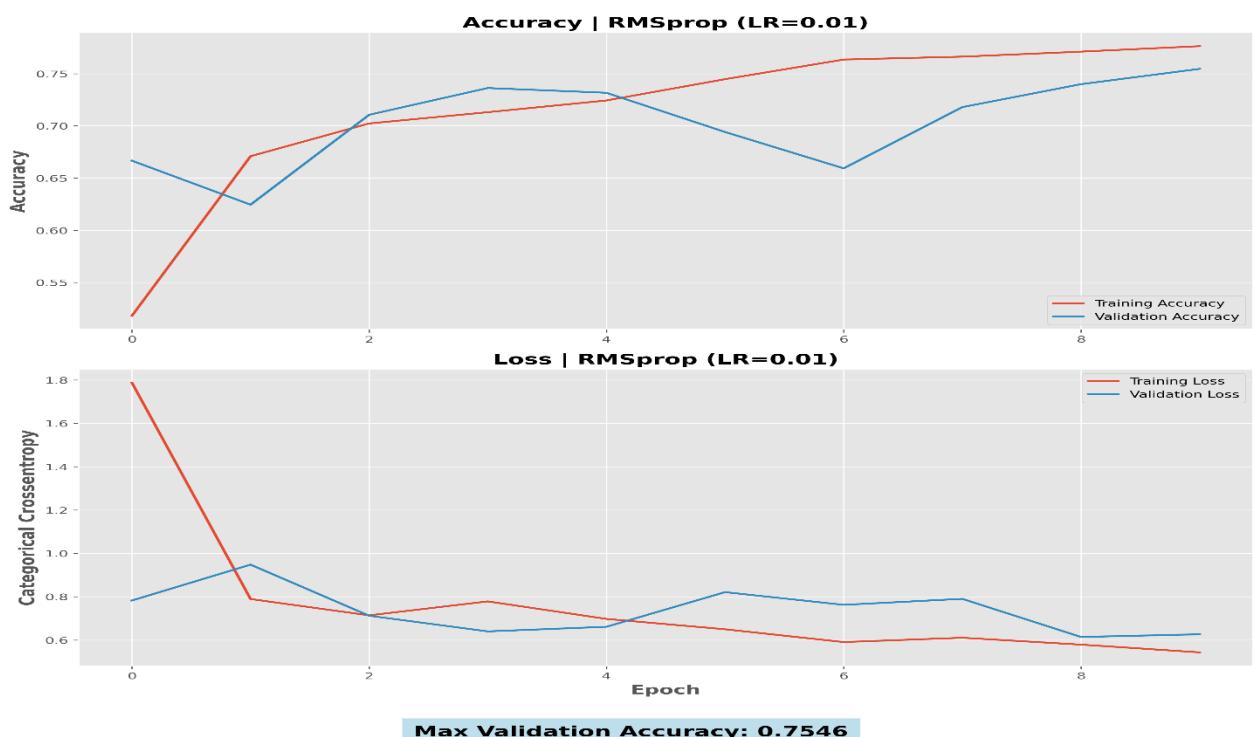
אופטימיזר שמתאים את קצב הלמידה לכל פרמטר בנפרד.

➤ RMSprop - Graph for learning rate 0.01:

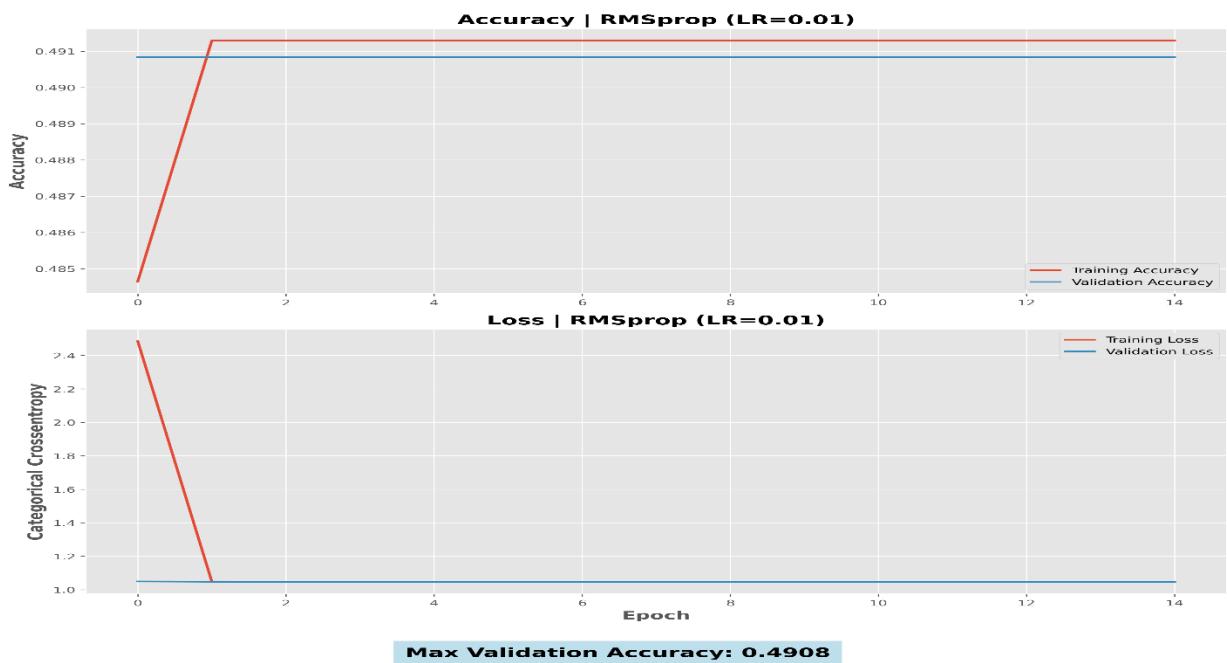
❖ Epochs 5:



❖ Epochs 10:

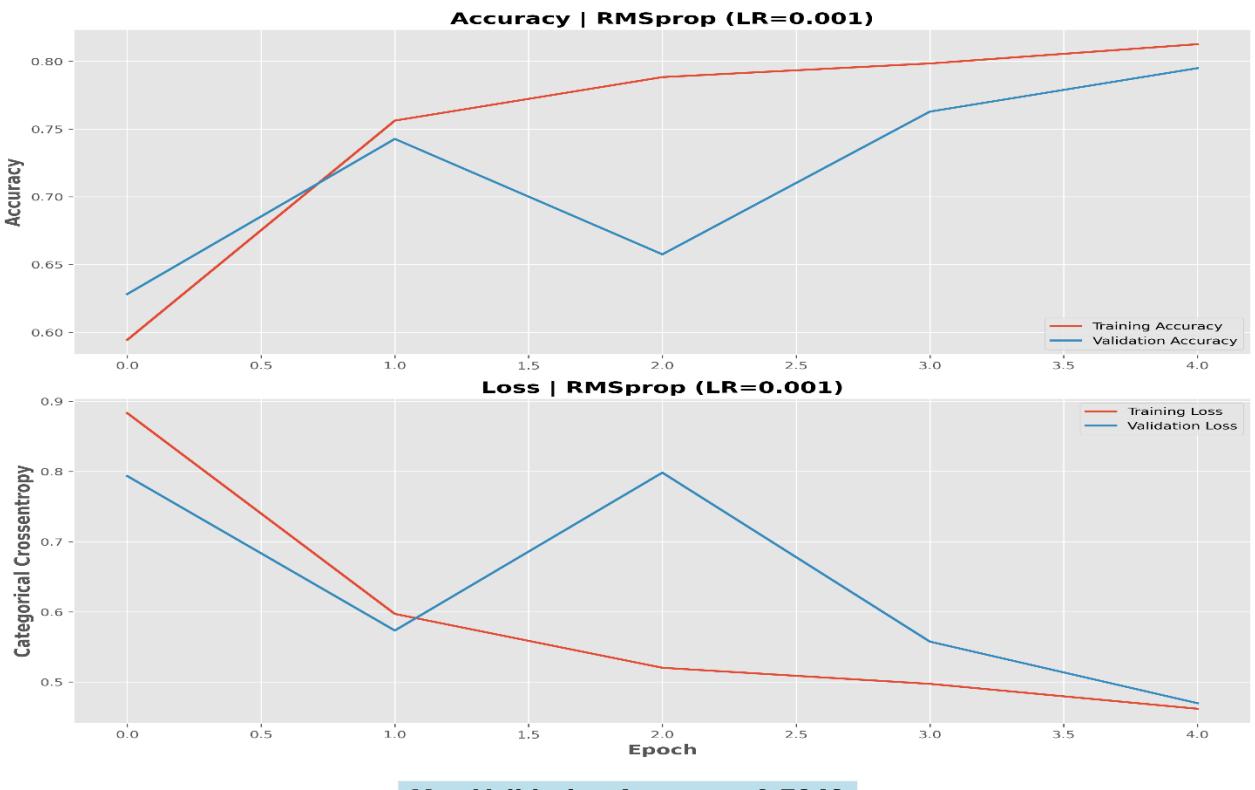


❖ Epochs 15:

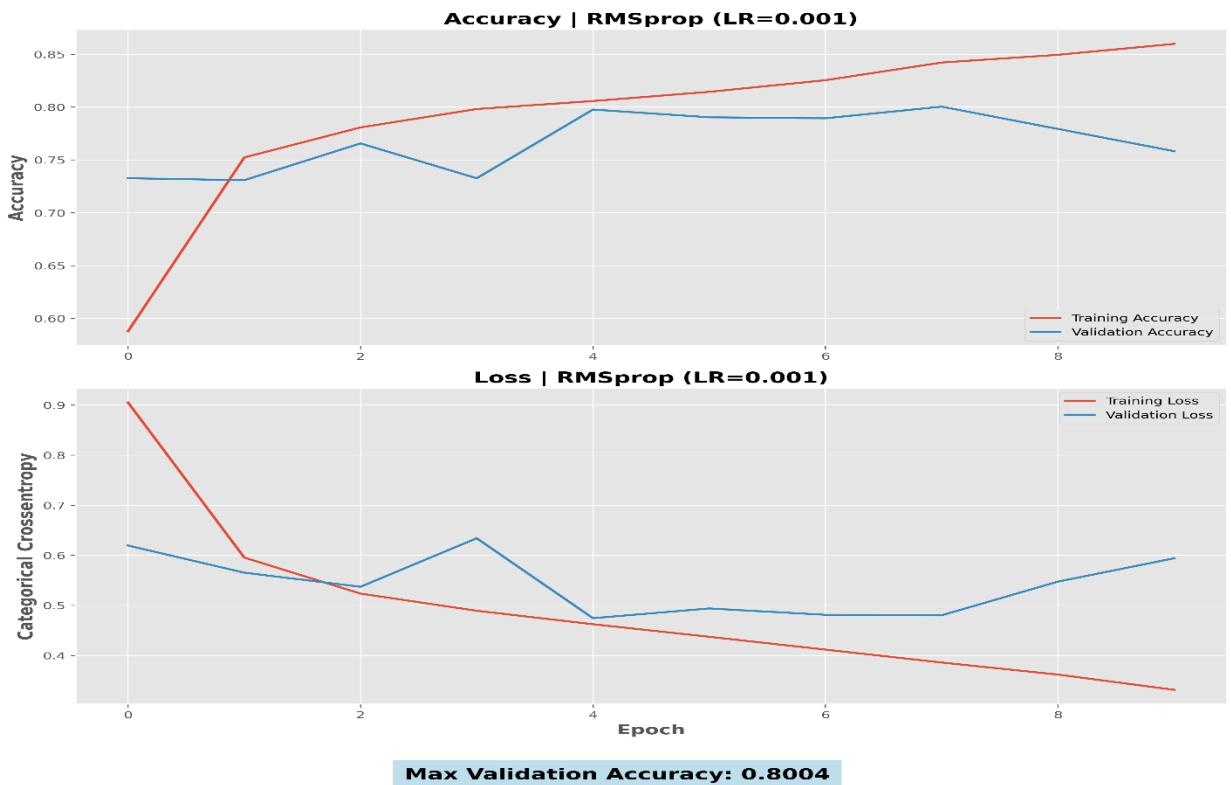


➤ RMSprop - Graph for learning rate 0.001:

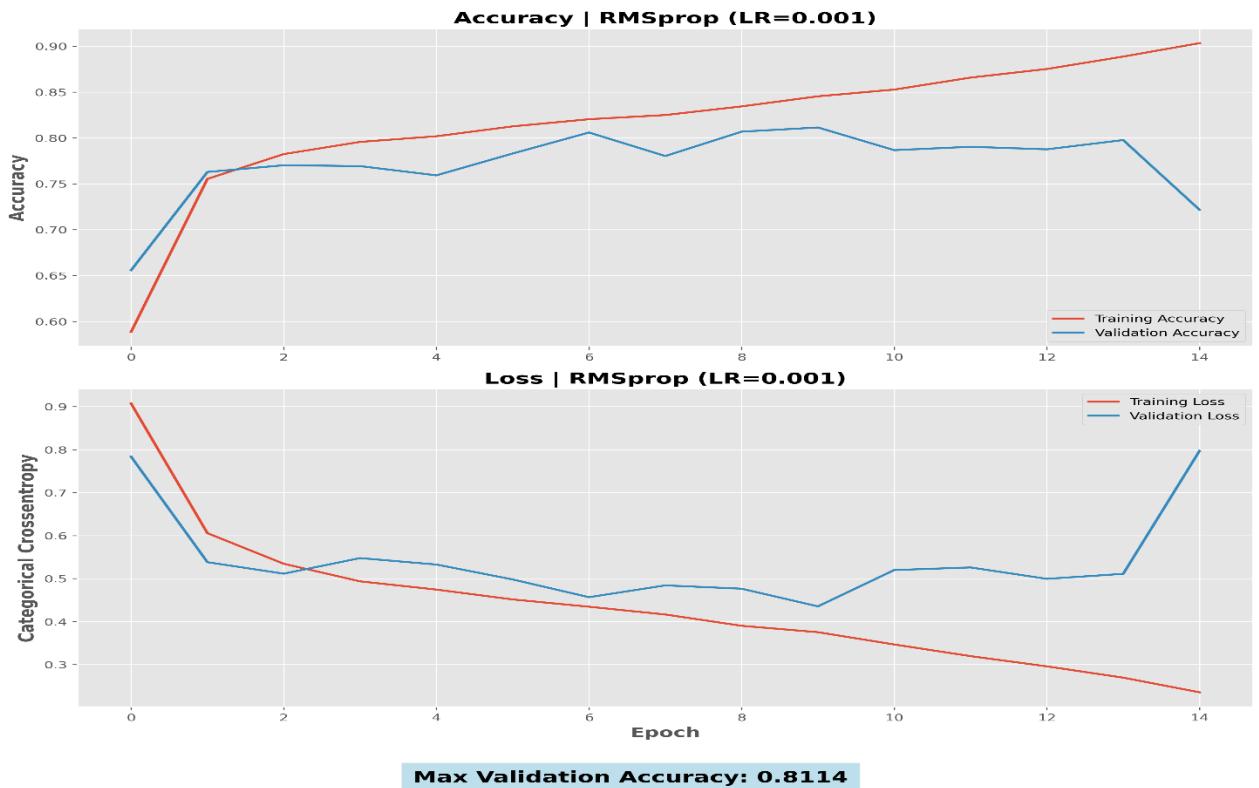
❖ Epochs 5:



❖ Epochs 10:

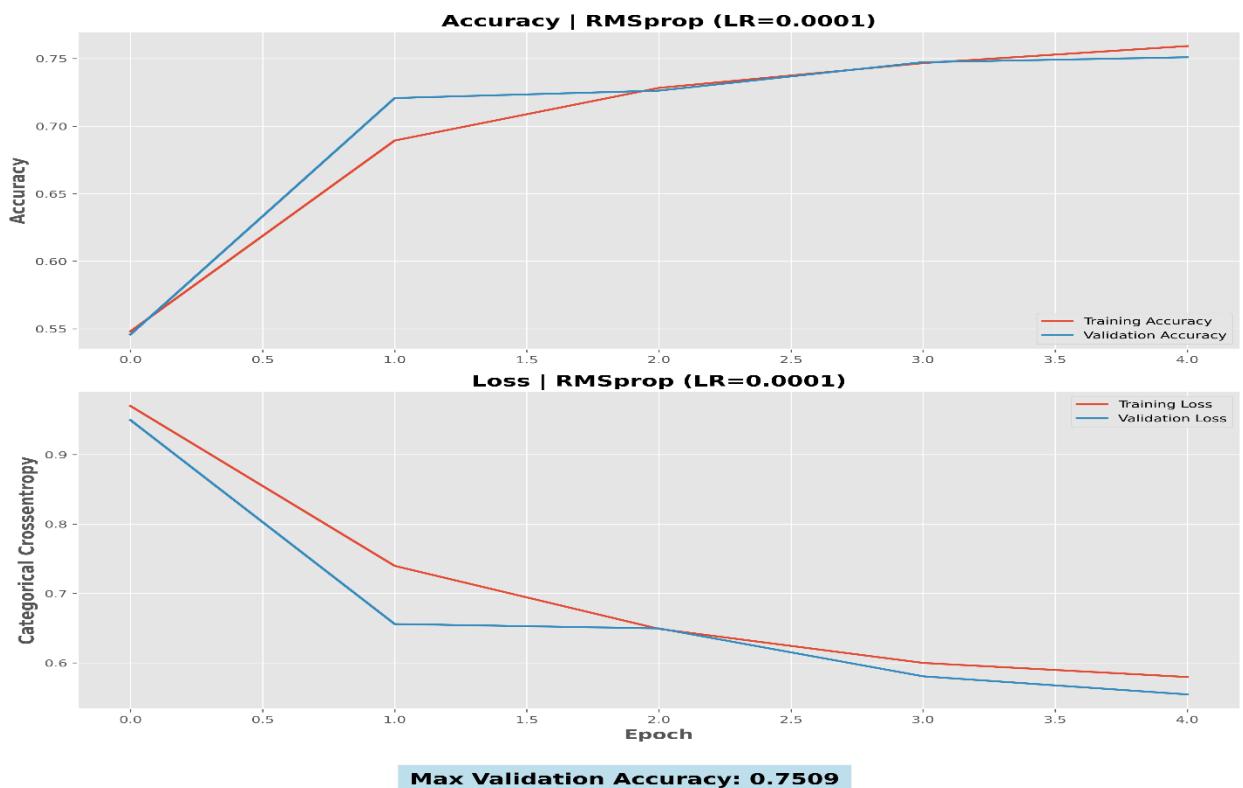


❖ Epochs 15:

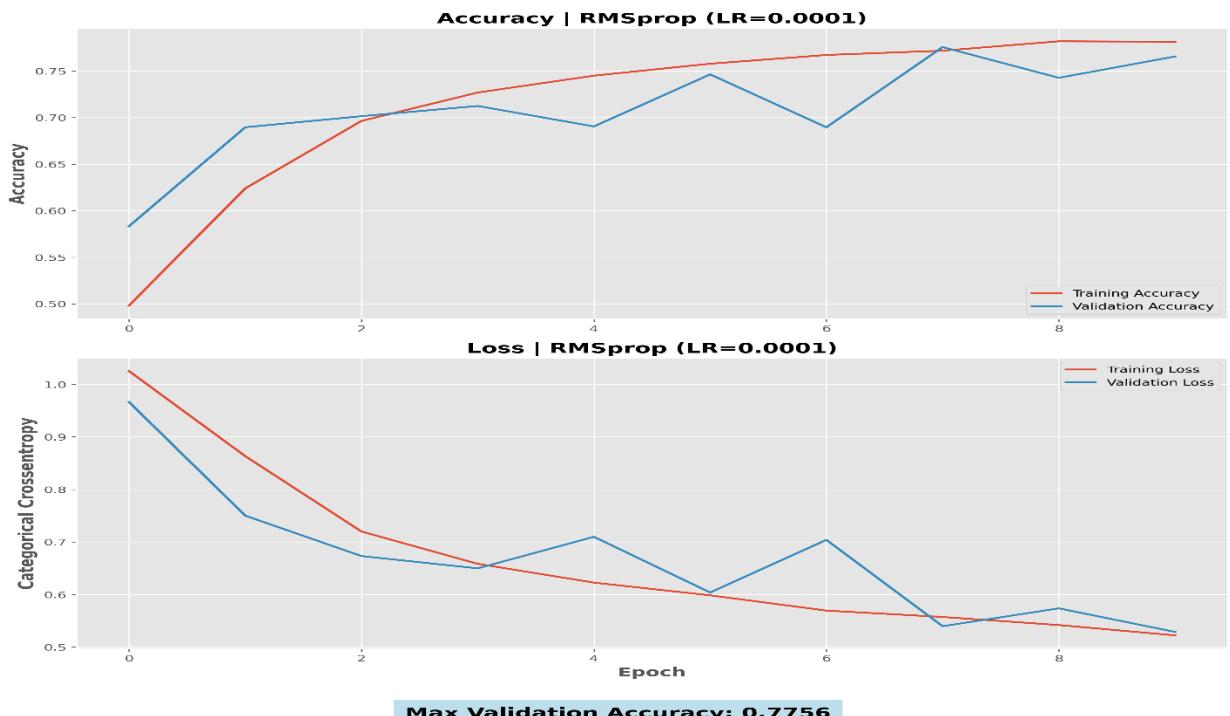


➤ RMSprop - Graph for learning rate 0.0001:

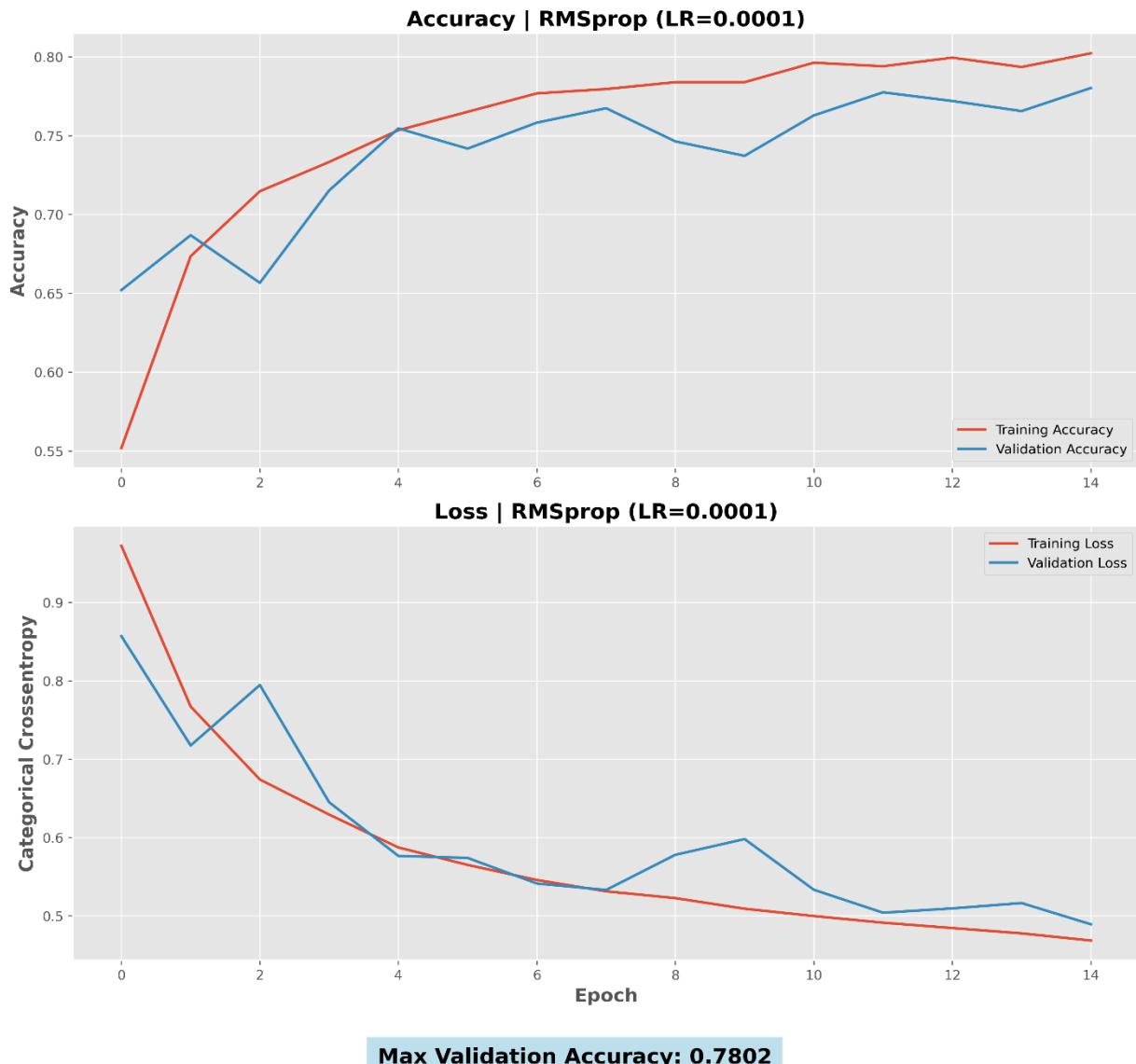
❖ Epochs 5:



❖ Epochs 10:



❖ Epochs 15:

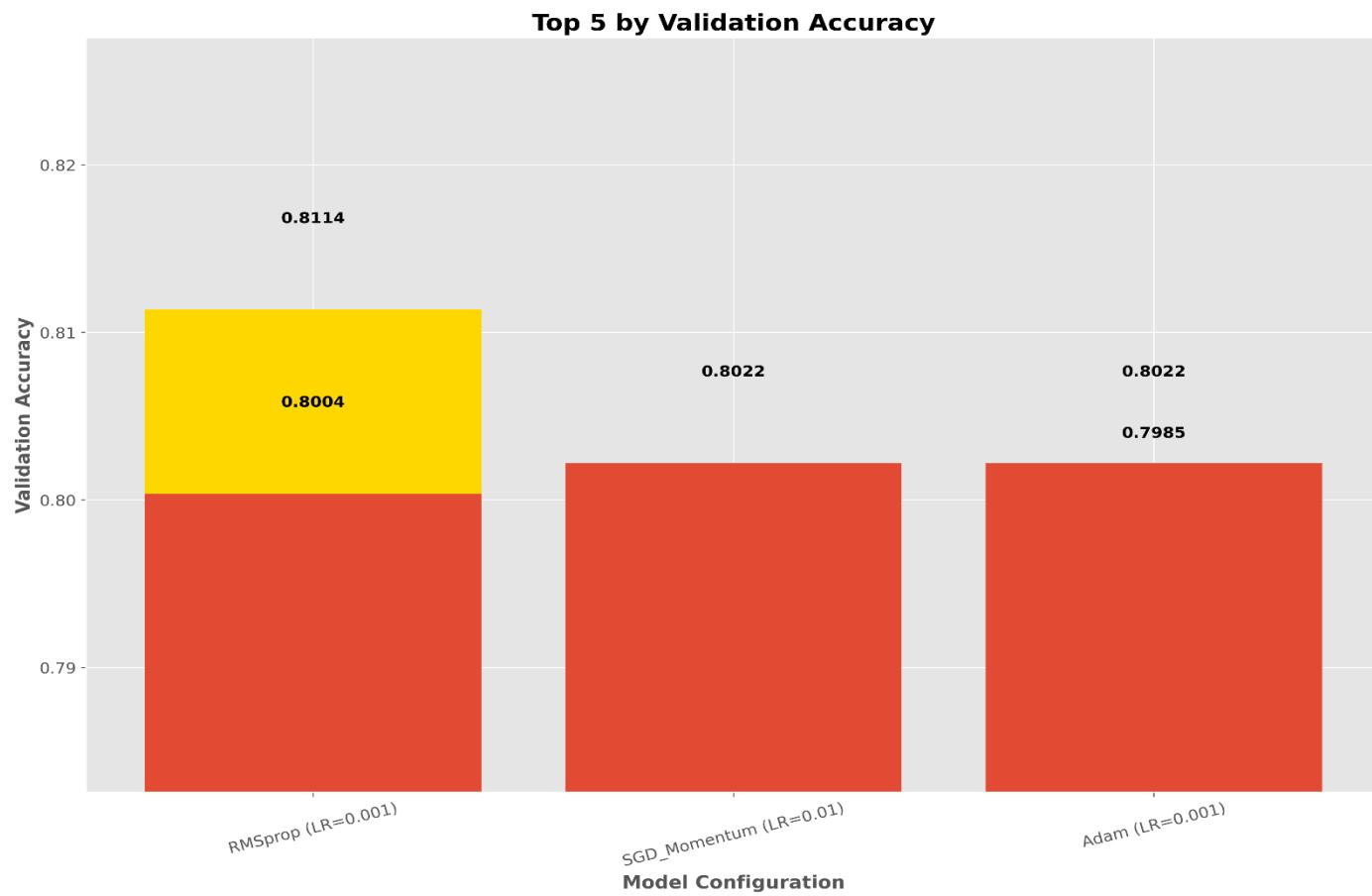


סיכום השוואתי בין כל הקונפיגורציות

לאחר סיום האימונים, בוצעה השוואת בין המודלים לפי דיקוק על סט הווילדציה.
ניתן לראות בגרף הסיכום כי הקונפיגורציה המובילה הייתה:

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

המודל הזה נבחר כמודל המנתח 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%)

סיווג תלת-קטgoriy Confusion Matrix

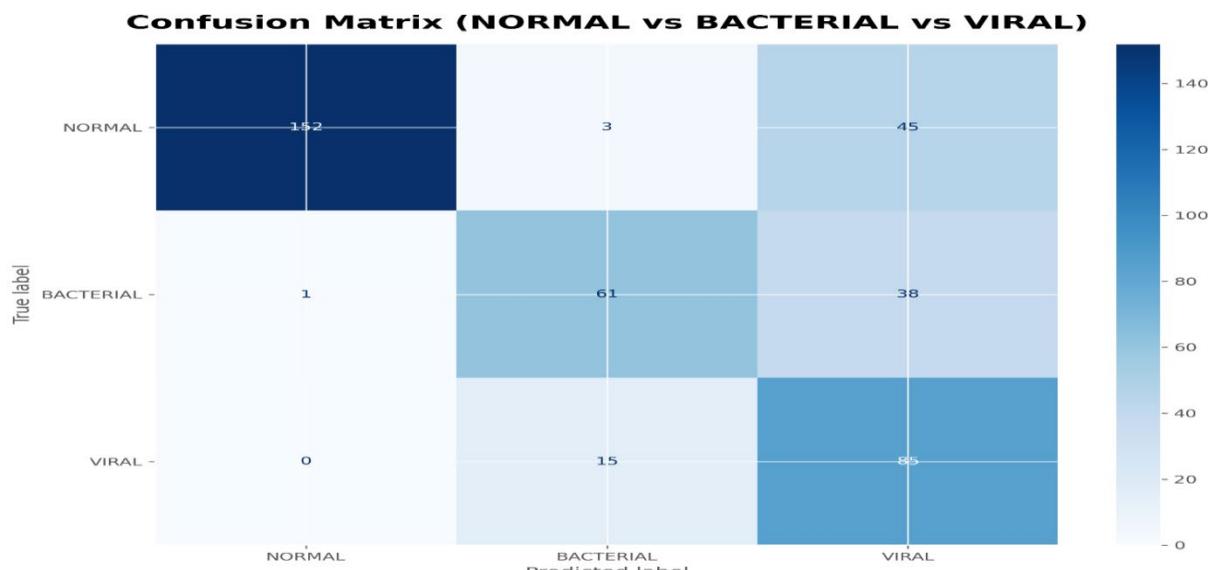
במשימה 4 אנו מסונגים את התמונה לשולש קטגוריות נפרדות Normal, Bacterial ו-Viral. תמונה ה- Pneumonia משוכנת באופן דטרמיניסטי לחת-הקטגוריות Bacterial או Viral על בסיס מילוט מפתח בשם הקובץ, כך שמתאפשרת תווית תלת-קטגורית עקבית בכל פיזיoli הנתונים. מטריצת הבלבול להלן מציגה את הביצועים על פני שלוש הקטגוריות הללו.

%81.14 : Accuracy

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

סיכום של משימה 4

- כל מודל אומן מהתחילה (לא בוצע Transfer Learning) עם שינוי בשכבה 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 הקונFIGורציות 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 ובמצאים השוואת שיטתיות.

סיכום כללי – דוח פרויקט: גילוי דלקת ריאות בצלומי רנטגן

במהלך הפרויקט פיתחנו והערכנו מודלים שונים מבוססי CNN לגילוי דלקת ריאות מסווגים שונים בצלומי רנטגן, תוך השוואת בין שיטות למידה عمוקה עם ולא Transfer Learning – בחינת אופטימיזרים שונים, השפעת פרמטרים היפר-פרמטריים, והרחבת הבעה לSieving רב-קטgoriy.

מהלכי העבודה המשותפים לכל המשימות

בכל המשימות הקפדנו לבצע חלוקה אחידה בין סט האימון, האימונות והבדיקה – כך שנשמרת עקביות והימנעות מ- Data Leakage. בנוסף, בוצעה הדמיה ויוזאלית של נתונים לצורך בדיקה ראשונית, ונמדדנו ממדדים עקובים כגון Accuracy, Precision, Recall ו- F1. כל רשות אומנה לאורץ טוחנים איחדים של Epochs ו- Learning Rate, והתקנו את תהליך ההשווואה באמצעות גרפים תואמים.

שיאי הביצועים (Highlights)

- המודל עם Fine Tuning ו- Transfer Learning (משימה 2) הציג את ה- Validation Accuracy הגבוה ביותר (**0.8946**) ו- Learning Rate ו- Epochs המאוזנת.
- במשימה 3 נמצא כי האופטימיזר Adam עם learning rate=0.001 הוביל את התוצאות הטובות ביותר, ולאחר הפעלת EarlyStopping התקבל חיסכון של 63.3% בזמן האימון ללא פגיעה משמעותית בדיקון.
- במשימה 4, לאחר התאמת הרשת לשיווג 3 קטגוריות והפעלה חוזרת של ניסויים כמו RMSprop + learning rate=0.001 + 15 Epochs סיימה את הביצועים הטובים ביותר, עם דיקוק ולידציה של **0.8114** ו- Confusion Matrix מובהק.

מסקנות מרכזיות

- מתאים במיוחד **Transfer Learning** מדויק כדי להימנע מ- Overfitting או Underfitting Fine Tuning.
- בחירה האופטימיזר היא קריטית – ורמות למידה מתונות הציגו תוצאות מובהקות ויציבות.
- EarlyStopping הוא כלייעיל למניעת Overfitting וליעול זמן האימון – במיוחד כשהוא משולב עם מנגנון למידה מתקדם.
- Sieving מרובה קטגוריות (Normal/Bacterial/Viral) אפשרי גם ללא Transfer Learning, כאשר נעשה שימוש נכון בשכבת Softmax ואופטימיזציה מותאמת.