```python
model.compile(
    optimizer=keras.optimizers.RMSprop(learning_rate=0.0005),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

RMSprop

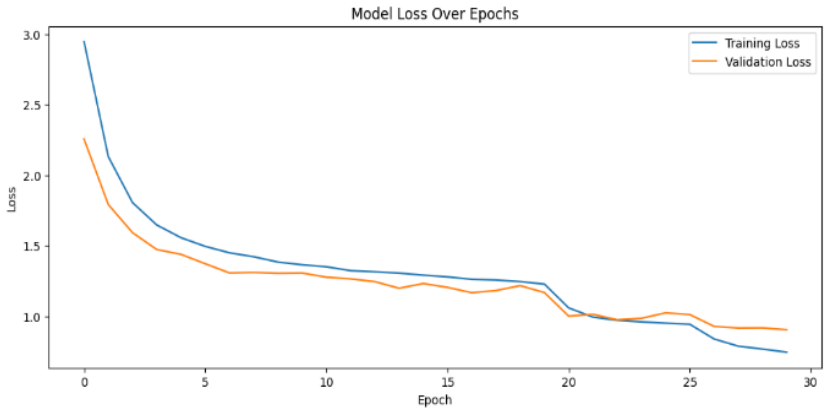
batch size=32_

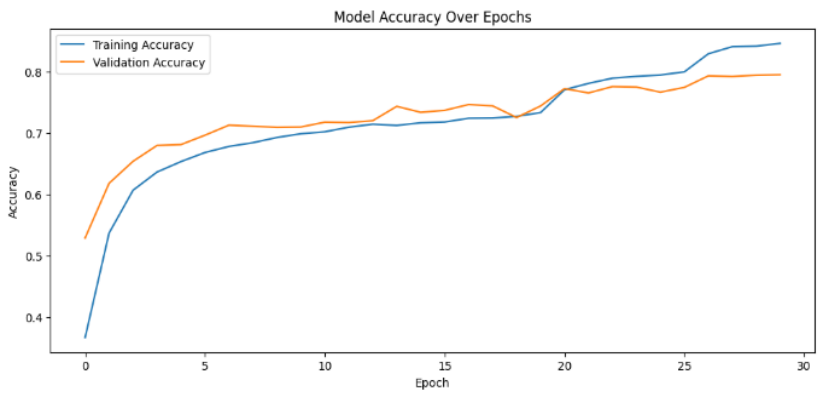
להלן המטריקות החשובות מתוך הריצה מ-Epoch 1 עד Epoch 30:

- **Epoch סופי (30):**
- Training Accuracy = **0.8553**
- Training Loss = **0.7190**
- Validation Accuracy = **0.7956**
- Validation Loss = **0.9029**
- Learning Rate = **1.25×10^{-4}**
- **Best Validation לאורך כל האימון:**
- best Val Accuracy = **0.7956** (ב-Epoch 30)
- best Val Loss = **0.9029** (ב-Epoch 30)
- **Test Evaluation (בסוף):**
- Test Accuracy = **0.7940**
- Test Loss = **0.9163**

בקיצור, המודל הגיע לכ-≈ 79.6 % וולידציה עם לוס ≈ 0.90, ו-≈ 79.4 % טסט עם לוס ≈ 0.92.

```python
he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
                ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
    layers.Dropout(0.3),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Dense(num_classes, activation='softmax')
])
```





```python
model.compile(
    optimizer=keras.optimizers.SGD(learning_rate=0.003, momentum=0.9),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

SGD+momentum

batch size=32_

להלן המטריקות החשובות מתוך הריצה מ־Epoch 1 עד Epoch 30:

- **Epoch סופי (30):**

Training Accuracy = 0.8531

Training Loss = 0.7654

Validation Accuracy = 0.7823

Validation Loss = 0.9931

Learning Rate = 3.75×10^{-4}

- **Best Validation לאורך כל האימון:**

Best Val Accuracy = 0.7867 (Epoch 28)

Best Val Loss = 0.9834 (Epoch 28)

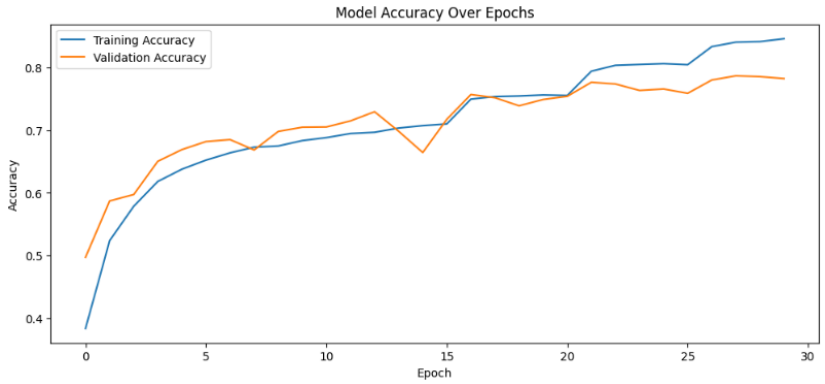
- **Test Evaluation (בסוף):**

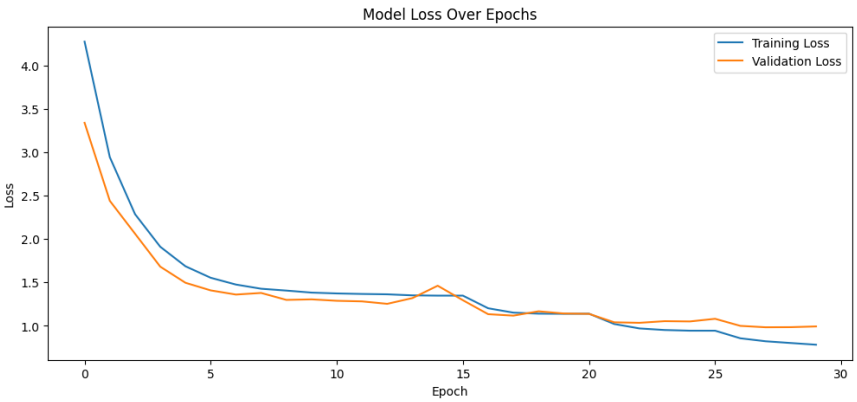
Test Accuracy = 0.7817

Test Loss = 1.0057

בקיצור, ה־Best Validation Accuracy הגיע לכ־78.7 % עם Loss≈0.98, ואילו על ה־Test קיבלנו כ־78.17 % עם Loss≈1.01.

```python
he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
                ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
                ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Dense(num_classes, activation='softmax')
])
```





```
model.compile(
    optimizer=keras.optimizers.SGD(learning_rate=0.003, momentum=0.9, nesterov=True
),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

NAG

batch size=32_

להלן המטריקות שחושבו מתוך הריצה (NAG, LR Schedule-) מ-Epoch 1 עד Epoch 30:

- **Epoch סופי (30)**

Training Accuracy = **0.8460**

Training Loss = **0.7593**

Validation Accuracy = **0.7789**

Validation Loss = **0.9922**

Learning Rate = $\mathbf{3.75 \times 10^{-4}}$

- **Best Validation לאורך כל האימון**

Best Val Accuracy = **0.7867** (Epoch 27)

Best Val Loss = **0.9614** (Epoch 27)

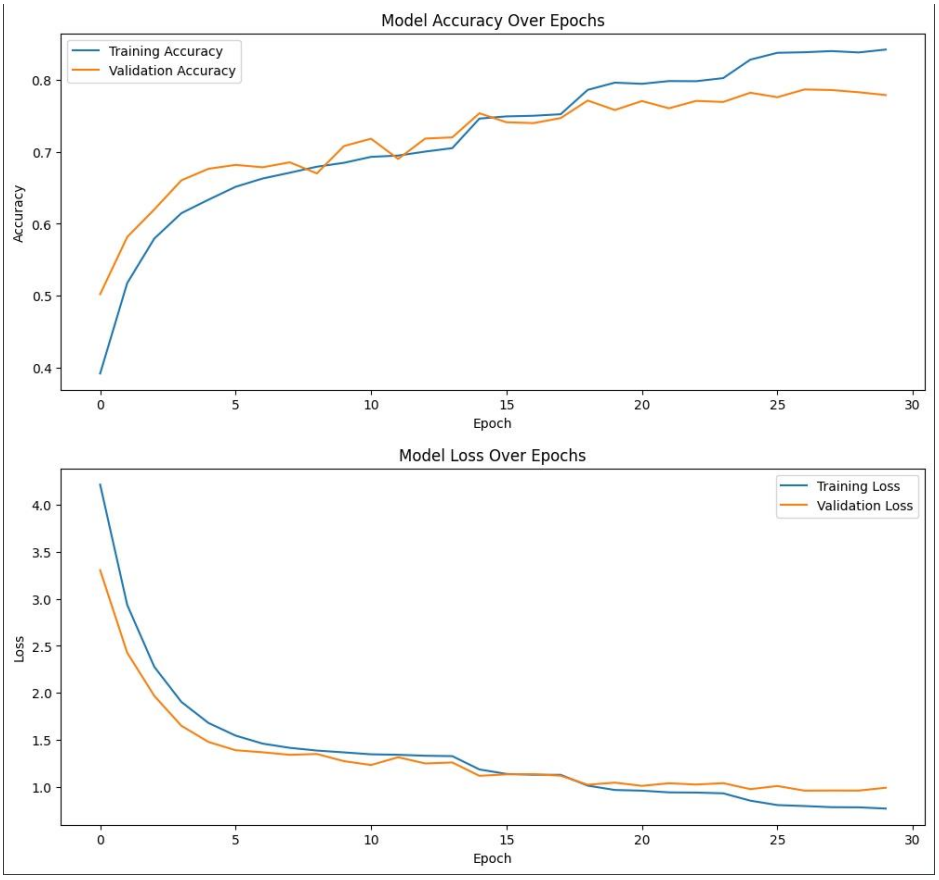
- **Test Evaluation (בסוף)**

Test Accuracy = **0.7812**

Test Loss = **0.9785**

בקיצור – המודל הגיע ל-78.7 ≈ Best Validation Accuracy % עם Loss ≈ 0.96, ואילו על ה-Test קיבלנו כ-78.1 % עם Loss ≈ 0.98, פער קטן שמרמז על התאמה טובה וללא אובר-פיטינג משמעותי.

```
[18] he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
            ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
            ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Dense(num_classes, activation='softmax')
])
```



```python
model.compile(
    optimizer=keras.optimizers.Adagrad(learning_rate=0.01),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

AdaGrad

batch size=32

להלן המטריקות שחושבו מתוך הריצה (Adagrad, LR=0.01) מ־Epoch 1 עד Epoch 30:

- **Epoch סופי (30):**

Training Accuracy = 0.8219

Training Loss = 1.0225

Validation Accuracy = 0.7591

Validation Loss = 1.1991

Learning Rate = 1×10^{-2}

- **Best Validation לאורך כל האימון:**

Best Val Accuracy = 0.7625 (Epoch 28)

Best Val Loss = 1.1982 (Epoch 27)

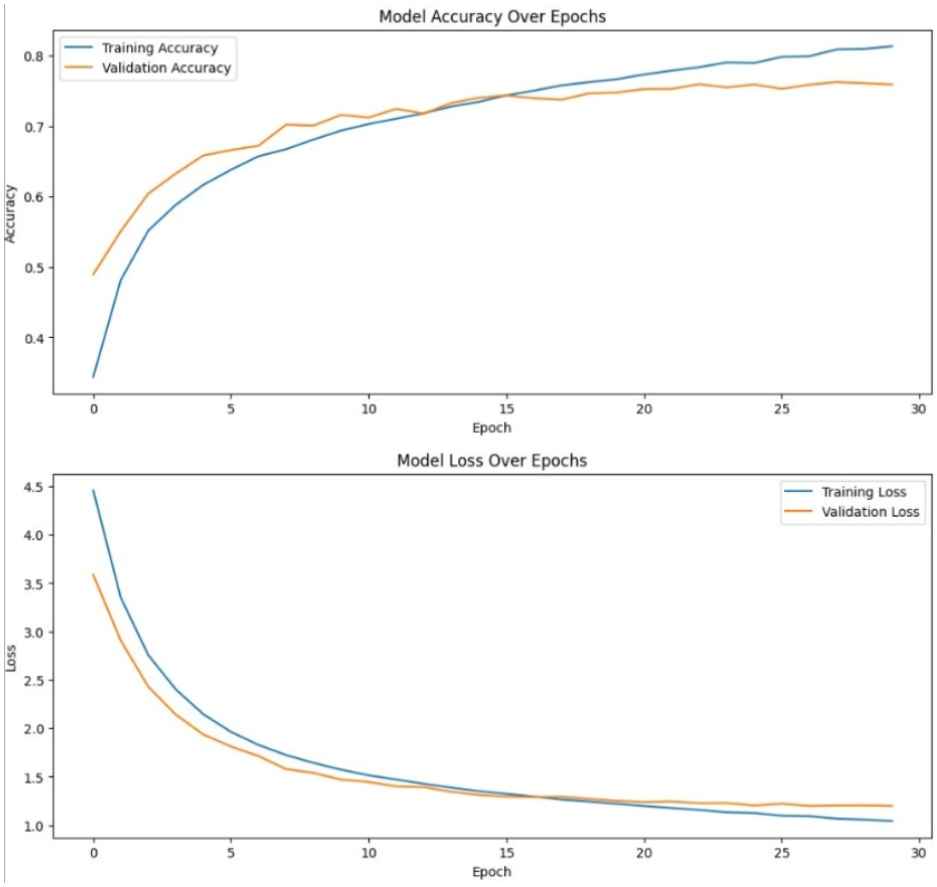
- **Test Evaluation (בסוף):**

Test Accuracy = 0.7616

Test Loss = 1.2092

בקיצור – המודל הגיע ל־76.25% ב־Val עם Loss≈1.20, ואילו על ה־Test קיבלנו כ־76.16% עם Loss≈1.21; התאמה סבירה וללא over-fitting משמעותי.

```python
[18] he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
            ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
            ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.005)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.6),

    layers.Dense(num_classes, activation='softmax')
])
```



```python
model.compile(
    optimizer=keras.optimizers.Adadelta(
        learning_rate=0.5
    ),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

AdaDelta

batch size=32

להלן המטריקות שחושבו מתוך הריצה (Adadelta, LR=0.5) מ-Epoch 1 עד Epoch 30:

- Epoch סופי (30):

Training Accuracy = 0.8095

Training Loss = 0.8889

Validation Accuracy = 0.7859

Validation Loss = 0.9328

Learning Rate = 1.25×10^{-1}

- Best Validation לאורך כל האימון:

Best Val Accuracy = 0.7859 (Epoch 30)

Best Val Loss = 0.9328 (Epoch 30)

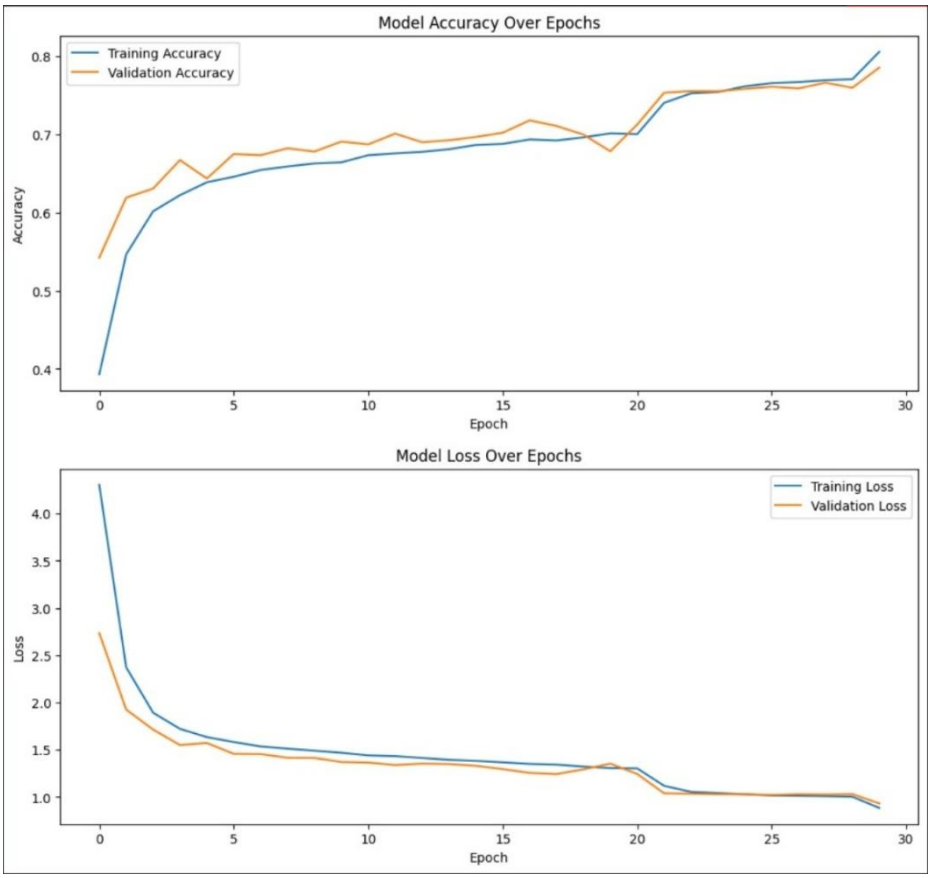
- Test Evaluation (בסוף):

Test Accuracy = 0.7772

Test Loss = 0.9466

בקיצור – המודל הגיע ל-78.59% ב-Val עם Loss 0.93≈, ואילו על ה-Test קיבלנו כ-77.72% עם Loss 0.95≈; התאמה טובה וללא over-fitting משמעותי.

```python
he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.004)
            ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.004)
            ),
    layers.BatchNormalization(),
    layers.Dropout(0.3),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.004)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.35),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.004)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.004)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.006)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.006)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.45),

    layers.Dense(num_classes, activation='softmax')
])
```



```python
model.compile(
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=5e-4,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-7
    ),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Adam

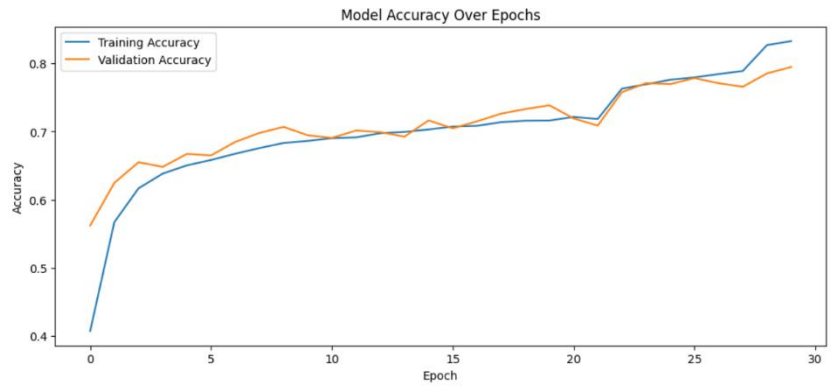
batch size=32

להלן המטריקות שחושבו מתוך הריצה מ-Epoch 1 עד Epoch 30 (Adam, LR התחלתי = 5×10^{-4}):

- **סופי (Epoch (30:**
  - Training Accuracy = 0.8389
  - Training Loss = 0.7868
  - Validation Accuracy = 0.7948
  - Validation Loss = 0.9226
  - Learning Rate = 1.25×10^{-4}
- **Best Validation לאורך כל האימון:**
  - Best Val Accuracy = 0.7948 (Epoch 30)
  - Best Val Loss = 0.9226 (Epoch 30)
- **Test Evaluation (בסוף):**
  - Test Accuracy = 0.7902
  - Test Loss = 0.9310

בקיצור – המודל הגיע ל-Validation Accuracy ≈79.5% ו-Test Accuracy ≈79.0%, עם Validation Loss יורד ל-0.92; התאמה טובה וללא over-fitting משמעותי.

```python
] he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
                ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
                ),
    layers.BatchNormalization(),
    layers.Dropout(0.3),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.35),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
        ),
    layers.BatchNormalization(),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.002)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.003)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Dense(num_classes, activation='softmax')
])
```



```python
model.compile(
    optimizer=tf.keras.optimizers.SGD(
        learning_rate=0.01
    ),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

SGD

batch size=32

להלן המטריקות החשובות שחושבו מתוך הריצה (SGD, LR=0.01) מ-Epoch 1 עד Epoch 30:

- Epoch סופי (30):
  - Training Accuracy = 0.8118
  - Training Loss = 0.8601
  - Validation Accuracy = 0.7516
  - Validation Loss = 1.0726
  - Learning Rate = 1×10^{-2}
- Best Validation לאורך כל האימון:
  - Best Val Accuracy = 0.7516 (Epoch 30)
  - Best Val Loss = 1.0726 (Epoch 30)
- Test Evaluation (בסוף):
  - Test Accuracy = 0.7429
  - Test Loss = 1.1062

```python
he_initializer = initializers.HeNormal()

model = keras.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.BatchNormalization(),

    layers.Conv2D(
        20, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
                ),
    layers.BatchNormalization(),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
                ),
    layers.BatchNormalization(),
    layers.Dropout(0.2),

    layers.Conv2D(
        68, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.3),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.4),

    layers.Conv2D(
        82, (3, 3), padding='same',
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.001)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Flatten(),

    layers.Dense(
        21,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.0005)
        ),
    layers.BatchNormalization(),

    layers.Dense(
        68,
        activation='leaky_relu',
        kernel_initializer=he_initializer,
        kernel_regularizer=regularizers.l2(0.0005)
        ),
    layers.BatchNormalization(),
    layers.Dropout(0.5),

    layers.Dense(num_classes, activation='softmax')
])
```

בקיצור, המודל הגיע לכ־75.16% דיוק בוולידציה ול־74.29% בדאטה של ה-Test, עם פער של כ־6% בין דיוק האימון לדיוק הוולידציה. נכון שהאימון יורד ללוס נמוך יחסית (0.86), אבל הוולידציה נותרת סביב 1.07, מה שמעיד על over-fitting מתון. ההתאמה בין וולידציה ל-Test טובה (פער ≈ 1%), ואין פה שיפור פתאומי שמעיד על רעש.