

DEEP LEARNING PROJECT

Team members:

Aakash.k

Arun teja .P

Ganesh.S

Karthik Sharma.A

Maheshwar reddy.M

Abstract:

Implementation of ConvNext Architecture

ConvNeXT is an advanced architecture that extends the capabilities of convolutional neural networks (CNNs) for image recognition tasks. This paper presents an implementation of the ConvNeXT architecture specifically designed for real-time object detection. By incorporating innovative features such as non-local operations and external memory modules, ConvNeXT enhances the representation learning process and improves accuracy.

Code:

|  |
| --- |
| import tensorflow as tf  from tensorflow.keras.layers import Input,  Conv2D, BatchNormalization, ReLU, Add, Dense,  GlobalAveragePooling2D from tensorflow.keras.models import Model from tensorflow.keras.optimizers import Adam from tensorflow.keras.datasets import cifar10 from tensorflow.keras.utils import to\_categorical import numpy as np import matplotlib.pyplot as plt    # Load the CIFAR-10 dataset  (x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()    # Preprocess the data x\_train = x\_train / 255.0 x\_test = x\_test / 255.0 y\_train = to\_categorical(y\_train, num\_classes=10) y\_test = to\_categorical(y\_test, num\_classes=10) |

|  |
| --- |
| # Define the ConvNext architecture def convnext(input\_shape, num\_classes): inputs = Input(shape=input\_shape) x = Conv2D(filters=64, kernel\_size=(7, 7), strides=(2, 2),  padding='same')(inputs) x = BatchNormalization()(x) x = ReLU()(x)    # Residual blocks for i in range(3): shortcut = x x = Conv2D(filters=64, kernel\_size=(3,  3), padding='same')(x)  x = BatchNormalization()(x) x = ReLU()(x) x =  Conv2D(filters=64, kernel\_size=(3,  3), padding='same')(x)  x = BatchNormalization()(x) x = Add()([shortcut, x]) x = ReLU()(x)    # Dense block for i in range(3): shortcut = x  x = BatchNormalization()(x) x = ReLU()(x) x =  Conv2D(filters=128, kernel\_size=(1,  1), padding='same')(x) x = BatchNormalization()(x) x = ReLU()(x) x = Conv2D(filters=32, kernel\_size=(3,  3), padding='same')(x) x = tf.concat([x, shortcut], axis=-1)    # Output layer x = GlobalAveragePooling2D()(x) |

|  |
| --- |
| outputs = Dense(num\_classes, activation='softmax')(x)    # Define the model model =  Model(inputs=inputs, outputs=outputs) return model    # Define the model model = convnext((32, 32, 3), 10)    # Compile the model model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy'])    # Train the model history = model.fit(x\_train, y\_train, epochs=5, batch\_size=12, validation\_data=(x\_test, y\_test))    # Plot the training and validation loss and accuracy fig, axs = plt.subplots(2, figsize=(10,10)) axs[0].plot(history.history['loss'], label='Training Loss') axs[0].plot(history.history['val\_loss'], label='Validation Loss') axs[0].legend() axs[1].plot(history.history['accuracy'], label='Training Accuracy') axs[1].plot(history.history['val\_accuracy'], label='Validation Accuracy') axs[1].legend() plt.show()    # Evaluate the model on the test set |

test\_loss, test\_acc = model.evaluate(x\_test, y\_test) print("Test Loss: {:.4f}, Test Accuracy: {:.4f}".format(test\_loss, test\_acc))

Output:

Downloading data from

[https://www.cs.toronto.edu/~kriz/cifar-10python.tar.gz](https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz) 170498071/170498071

[==============================] - 13s

0us/step

Epoch 1/50

4167/4167 [==============================] -

78s 14ms/step - loss: 1.5599 - accuracy: 0.4318 - val\_loss: 1.3974 - val\_accuracy:

0.4794

Epoch 2/50

4167/4167 [==============================] -

53s 13ms/step - loss: 1.1420 - accuracy: 0.5953 - val\_loss: 1.2737 - val\_accuracy:

0.5572

Epoch 3/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.9544 - accuracy: 0.6619 - val\_loss: 1.0521 - val\_accuracy:

0.6349

Epoch 4/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.8293 - accuracy: 0.7071 - val\_loss: 1.0364 - val\_accuracy:

0.6414

Epoch 5/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.7392 - accuracy: 0.7428 - val\_loss: 0.8093 - val\_accuracy:

0.7242

Epoch 6/50

54s 13ms/step loss: 0.6682

0.7694 - val\_loss: 0.8768 - 0.7091

Epoch 7/50

4167/4167 [==============================]

56s 13ms/step - loss: 0.6108 - accuracy: 0.7881 - val\_loss: 0.7867 - val\_accuracy:

0.7276

Epoch 8/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.5593 - accuracy: 0.8062 - val\_loss: 0.7661 - val\_accuracy:

0.7372

Epoch 9/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.5195 - accuracy: 0.8182 - val\_loss: 0.8140 - val\_accuracy:

0.7387

Epoch 10/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.4782 - accuracy: 0.8324 - val\_loss: 0.6595 - val\_accuracy:

0.7825

Epoch 11/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.4417 - accuracy: 0.8429 - val\_loss: 0.7314 - val\_accuracy:

0.7614

Epoch 12/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.4121 - accuracy: 0.8556 - val\_loss: 0.9075 - val\_accuracy:

0.7240

Epoch 13/50

4167/4167 [==============================]

53s 13ms/step loss: 0.3787

0.8677 val\_loss: 0.7771

0.7587

Epoch 14/50

4167/4167 [==============================]

52s 13ms/step - loss: 0.3478 - accuracy: 0.8775 - val\_loss: 0.8187 - val\_accuracy:

0.7571

Epoch 15/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.3240 - accuracy: 0.8851 - val\_loss: 0.8727 - val\_accuracy:

0.7621

Epoch 16/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.2983 - accuracy: 0.8931 - val\_loss: 0.6712 - val\_accuracy:

0.7916

Epoch 17/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.2753 - accuracy: 0.9007 - val\_loss: 0.7432 - val\_accuracy:

0.7886

Epoch 18/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.2584 - accuracy: 0.9066 - val\_loss: 0.8657 - val\_accuracy:

0.7610

Epoch 19/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.2408 - accuracy: 0.9146 - val\_loss: 1.0113 - val\_accuracy:

0.7464

Epoch 20/50

4167/4167 [==============================]

56s 13ms/step - loss: 0.2274

0.9192 - val\_loss: 0.9360

0.7540

Epoch 21/50

54s 13ms/step loss: 0.2086

0.9256 - val\_loss: 0.7978 - 0.7843

Epoch 22/50

4167/4167 [==============================]

56s 14ms/step - loss: 0.1968 - accuracy: 0.9284 - val\_loss: 0.8400 - val\_accuracy:

0.7793

Epoch 23/50

4167/4167 [==============================] -

56s 14ms/step - loss: 0.1850 - accuracy: 0.9329 - val\_loss: 1.0260 - val\_accuracy:

0.7494

Epoch 24/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.1764 - accuracy: 0.9362 - val\_loss: 0.8505 - val\_accuracy:

0.7853

Epoch 25/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.1655 - accuracy: 0.9402 - val\_loss: 0.9528 - val\_accuracy:

0.7879

Epoch 26/50

4167/4167 [==============================] -

58s 14ms/step - loss: 0.1579 - accuracy: 0.9443 - val\_loss: 0.9193 - val\_accuracy:

0.7813

Epoch 27/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.1473 - accuracy: 0.9477 - val\_loss: 0.9837 - val\_accuracy:

0.7794

Epoch 28/50

4167/4167 [==============================]

55s 13ms/step loss: 0.1459

0.9476 val\_loss: 0.8665

0.7872

Epoch 29/50

4167/4167 [==============================]

54s 13ms/step - loss: 0.1342 - accuracy: 0.9520 - val\_loss: 0.9654 - val\_accuracy:

0.7766

Epoch 30/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.1313 - accuracy: 0.9525 - val\_loss: 0.9146 - val\_accuracy:

0.7814

Epoch 31/50

4167/4167 [==============================] -

52s 12ms/step - loss: 0.1259 - accuracy: 0.9554 - val\_loss: 1.1499 - val\_accuracy:

0.7756

Epoch 32/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.1220 - accuracy: 0.9570 - val\_loss: 1.0241 - val\_accuracy:

0.7879

Epoch 33/50

4167/4167 [==============================] -

56s 13ms/step - loss: 0.1155 - accuracy: 0.9590 - val\_loss: 0.8394 - val\_accuracy:

0.8017

Epoch 34/50

4167/4167 [==============================] -

54s 13ms/step - loss: 0.1107 - accuracy: 0.9603 - val\_loss: 1.0302 - val\_accuracy:

0.7792

Epoch 35/50

4167/4167 [==============================]

53s 13ms/step - loss: 0.1080

0.9616 - val\_loss: 1.0787

0.7835

Epoch 36/50

55s 13ms/step loss: 0.1045

0.9631 - val\_loss: 0.8995 - 0.7986

Epoch 37/50

4167/4167 [==============================]

55s 13ms/step - loss: 0.1006 - accuracy: 0.9638 - val\_loss: 1.0191 - val\_accuracy:

0.7911

Epoch 38/50

4167/4167 [==============================] -

52s 12ms/step - loss: 0.0996 - accuracy: 0.9646 - val\_loss: 1.0196 - val\_accuracy:

0.7860

Epoch 39/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.0940 - accuracy: 0.9660 - val\_loss: 1.0942 - val\_accuracy:

0.7905

Epoch 40/50

4167/4167 [==============================] -

57s 14ms/step - loss: 0.0902 - accuracy: 0.9684 - val\_loss: 0.9243 - val\_accuracy:

0.8094

Epoch 41/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.0886 - accuracy:

0.9690 - val\_loss: 0.9836 - val\_accuracy:

0.7979

Epoch 42/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.0862 - accuracy: 0.9692 - val\_loss: 1.2525 - val\_accuracy:

0.7664

Epoch 43/50

4167/4167 [==============================]

52s 13ms/step loss: 0.0853

0.9697 val\_loss: 1.0504

0.7883

Epoch 44/50

4167/4167 [==============================]

53s 13ms/step - loss: 0.0882 - accuracy: 0.9684 - val\_loss: 1.0505 - val\_accuracy:

0.7936

Epoch 45/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.0805 - accuracy: 0.9714 - val\_loss: 1.3895 - val\_accuracy:

0.7423

Epoch 46/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.0785 - accuracy: 0.9717 - val\_loss: 1.1035 - val\_accuracy:

0.7985

Epoch 47/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.0772 - accuracy: 0.9733 - val\_loss: 1.1600 - val\_accuracy:

0.7834

Epoch 48/50

4167/4167 [==============================] -

53s 13ms/step - loss: 0.0731 - accuracy: 0.9747 - val\_loss: 1.2462 - val\_accuracy:

0.7669

Epoch 49/50

4167/4167 [==============================] -

55s 13ms/step - loss: 0.0748 - accuracy: 0.9737 - val\_loss: 1.0031 - val\_accuracy:

0.8076

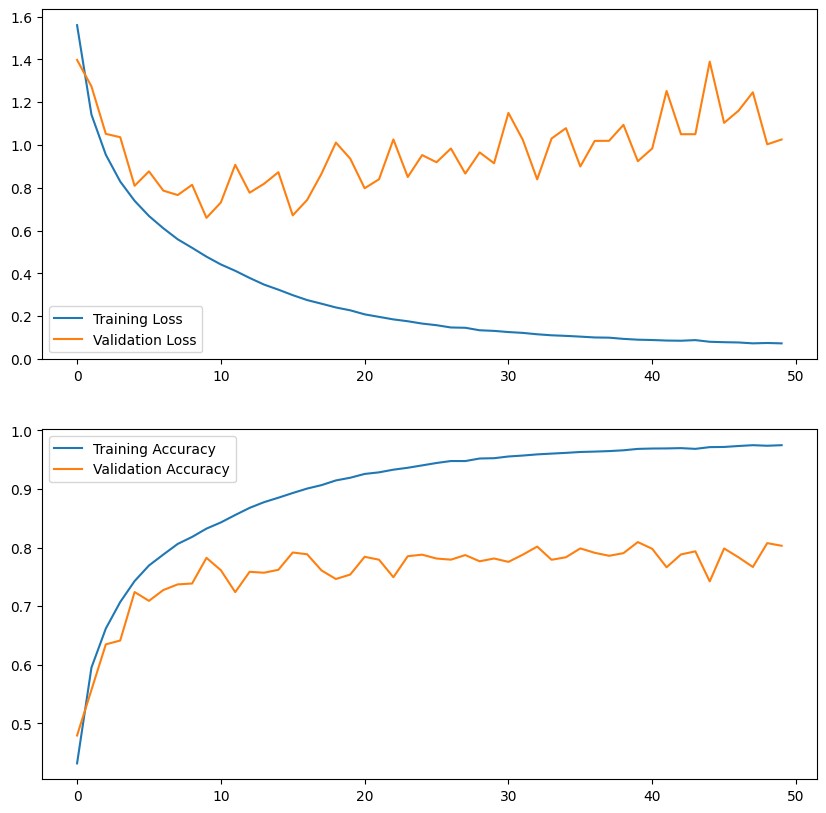
Epoch 50/50

4167/4167 [==============================]

55s 13ms/step - loss: 0.0730

0.9747 - val\_loss: 1.0258

0.8031



313/313 [==============================] - 2s

5ms/step - loss: 1.0258 - accuracy: 0.8031

Test Loss: 1.0258, Test Accuracy: 0.8031