List of Programs:

1. Create Tensors and perform basic operations with tensors
2. Create Tensors and apply split & merge operations and statistics operations.
3. Design single unit perception for classification of iris dataset without using predefined models
4. Design, train and test the MLP for tabular data and verify various activation functions and optimizers tensor flow.
5. Design and implement to classify 32x32 images using MLP using tensor flow/keras and check the accuracy.
6. Design and implement a simple RNN model with tensor flow / keras and check accuracy.
7. Design and implement LSTM model with tensor flow / keras and check accuracy.
8. Design and implement GRU model with tensor flow / keras and check accuracy.
9. Design and implement a CNN model to classify multi category JPG images with tensor flow / keras and check accuracy. Predict labels for new images.
10. Design and implement a CNN model to classify multi category tiff images with tensorf low / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit like regulizers, dropouts etc.
11. Implement a CNN architecture (LeNet, Alexnet, VGG, etc) model to classify multi category Satellite images with tensor flow / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit.
12. Implement an Auto encoder to de-noise image.
13. Implement a GAN application to convert images

In [4]:
```
# ==================== 1. TENSOR BASIC OPERATIONS ====================
import tensorflow as tf
# Create tensors
a = tf.constant([1, 2, 3])
b = tf.constant([4, 5, 6])
print("Tensor a:", a.numpy())
print("Tensor b:", b.numpy())
print("Add:", tf.add(a, b).numpy())
print("Multiply:", tf.multiply(a, b).numpy())
print("Mean:", tf.reduce_mean(a).numpy())
```

```
Tensor a: [1 2 3]
Tensor b: [4 5 6]
Add: [5 7 9]
Multiply: [ 4 10 18]
Mean: 2
```

In [16]:
```
 #==================== 2. TENSOR SPLIT, MERGE & STATISTICS ===================

import tensorflow as tf
# Create tensor
t = tf.constant([[1, 2, 3, 4], [5, 6, 7, 8]])
print("Original:", t)
# Split
split = tf.split(t, 2, axis=1)
print("Split:", split)
# Merge
merged = tf.concat(split, axis=1)
print("Merged:", merged)
# Statistics
print("Mean:", tf.reduce_mean(t).numpy())
print("Max:", tf.reduce_max(t).numpy())
print("Min:", tf.reduce_min(t).numpy())
```

```
Original: tf.Tensor(
[[1 2 3 4]
 [5 6 7 8]], shape=(2, 4), dtype=int32)
Split: [<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[1, 2],
       [5, 6]], dtype=int32)>, <tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[3, 4],
       [7, 8]], dtype=int32)>]
Merged: tf.Tensor(
[[1 2 3 4]
 [5 6 7 8]], shape=(2, 4), dtype=int32)
Mean: 4
Max: 8
Min: 1
```

In [17]:
```
# ==================== 3. PERCEPTRON FOR IRIS ====================
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data[:, :2]
y = (iris.target == 0).astype(int)

W = np.array([[1.0], [-1.0]])
b = 0.2
z = X @ W + b

ypred = (z > 0).astype(int)
print( "First 10 True labels (Setosa=1, Others=0):", y[:10])
print("First 10 Predicted outputs:",ypred[:10].flatten())
```

```
First 10 True labels (Setosa=1, Others=0): [1 1 1 1 1 1 1 1 1 1]
First 10 Predicted outputs: [1 1 1 1 1 1 1 1 1 1]
```

In [19]:
```
# ==================== 4. MLP FOR TABULAR DATA ====================
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

X = load_iris().data
y = load_iris().target

xtrain,xtest,ytrain,ytest= train_test_split(X,y,test_size = 0.2)

activations = ["relu", "sigmoid", "tanh"]
optimizers = ["sgd", "adam", "rmsprop"]

results = []

for act in activations:
  for opt in optimizers:
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(32,activation = act),
        tf.keras.layers.Dense(16,activation=act),
        tf.keras.layers.Dense(3, activation = "softmax")
    ])
    model.compile(optimizer = opt, loss = 'sparse_categorical_crossentropy',me
    model.fit(xtrain,ytrain,epochs = 100, batch_size = 8,verbose = 0)
    loss, acc = model.evaluate(xtest,ytest)
    results.append((act, loss, acc))
    print("training with activation",act," optimizer: ",opt)
    print("testing loss ", loss," testing accuracy= ", acc)

print("\nSummary of Accuracies:")
for act, opt, acc in results:
  print(f"Activation={act} | Optimizer={opt} | Accuracy={acc:.4f}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 505ms/step - accuracy: 0.9000 - loss: 0.2952
training with activation relu  optimizer:  sgd
testing loss  0.29518696665763855  testing accuracy=  0.8999999761581421
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 583ms/step - accuracy: 0.9333 - loss: 0.1830
training with activation relu  optimizer:  adam
testing loss  0.18301525712013245  testing accuracy=  0.9333333373069763
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 395ms/step - accuracy: 0.9000 - loss: 0.2015
training with activation relu  optimizer:  rmsprop
testing loss  0.20153045654296875  testing accuracy=  0.8999999761581421
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 376ms/step - accuracy: 0.5667 - loss: 0.8356
training with activation sigmoid  optimizer:  sgd
testing loss  0.8355898261070251  testing accuracy=  0.5666666626930237
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 380ms/step - accuracy: 0.9000 - loss: 0.1957
training with activation sigmoid  optimizer:  adam
testing loss  0.19565658271312714  testing accuracy=  0.8999999761581421
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 718ms/step - accuracy: 0.9333 - loss: 0.1695
training with activation sigmoid  optimizer:  rmsprop
testing loss  0.1695079207420349  testing accuracy=  0.9333333373069763
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 386ms/step - accuracy: 0.9000 - loss: 0.2758
training with activation tanh  optimizer:  sgd
testing loss  0.2758440673351288  testing accuracy=  0.8999999761581421
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 379ms/step - accuracy: 0.9333 - loss: 0.1806
training with activation tanh  optimizer:  adam
testing loss  0.1805601418018341  testing accuracy=  0.9333333373069763
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 380ms/step - accuracy: 0.9333 - loss: 0.2076
training with activation tanh  optimizer:  rmsprop
testing loss  0.20755943655967712  testing accuracy=  0.9333333373069763

Summary of Accuracies:
Activation=relu | Optimizer=0.29518696665763855 | Accuracy=0.9000
Activation=relu | Optimizer=0.18301525712013245 | Accuracy=0.9333
Activation=relu | Optimizer=0.20153045654296875 | Accuracy=0.9000
Activation=sigmoid | Optimizer=0.8355898261070251 | Accuracy=0.5667
Activation=sigmoid | Optimizer=0.19565658271312714 | Accuracy=0.9000
Activation=sigmoid | Optimizer=0.1695079207420349 | Accuracy=0.9333
Activation=tanh | Optimizer=0.2758440673351288 | Accuracy=0.9000
Activation=tanh | Optimizer=0.1805601418018341 | Accuracy=0.9333
Activation=tanh | Optimizer=0.20755943655967712 | Accuracy=0.9333
```

In [20]:
```python
# ==================== 5. MLP FOR 32x32 IMAGES (CIFAR-10) ====================
import tensorflow as tf
import pickle
import numpy as np
from sklearn.model_selection import train_test_split

with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding="bytes")

X = batch[b"data"]
y = np.array(batch[b"labels"])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

X_train = X_train / 255.0
X_test  = X_test  / 255.0
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(10, activation="softmax")
])

model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=

model.fit(X_train, y_train, epochs=5, batch_size=128, verbose=1)

loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Accuracy: {acc*100:.2f}%")


model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)


model.fit(X_train, y_train, epochs=5, batch_size=128, verbose=1)

loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Accuracy: {acc*100:.2f}%")
```

```
Epoch 1/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 0.1632 - loss: 2.2679
Epoch 2/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2894 - loss: 1.9544
Epoch 3/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3048 - loss: 1.9209
Epoch 4/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3426 - loss: 1.8231
Epoch 5/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3700 - loss: 1.7740
Accuracy: 38.90%
Epoch 1/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.3553 - loss: 1.8103
Epoch 2/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.3942 - loss: 1.7046
Epoch 3/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4156 - loss: 1.6717
Epoch 4/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3981 - loss: 1.6801
Epoch 5/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4049 - loss: 1.6603
Accuracy: 40.45%
```

In [21]:
```
# ==================== 6. SIMPLE RNN ====================

import tensorflow as tf
import pandas as pd
```

```python
# Load + prepare data
df = pd.read_csv("netflix_titles.csv.csv").dropna(subset=["description"])
df["label"] = (df["type"] == "Movie").astype(int)

tokenizer = tf.keras.preprocessing.text.Tokenizer(char_level=True)
tokenizer.fit_on_texts(df["description"])
X = tokenizer.texts_to_sequences(df["description"])
X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=100).reshape(-1,10
y = df["label"].values

# RNN Model
model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(32, input_shape=(100,1)),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy
model.fit(X, y, epochs=5, verbose=1)

# Accuracy
print("Accuracy:", model.evaluate(X, y, verbose=0)[1]* 100)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWa
rning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
Epoch 1/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 6s 10ms/step - accuracy: 0.6388 - loss: 0.6491
Epoch 2/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6948 - loss: 0.6164
Epoch 3/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.7074 - loss: 0.6050
Epoch 4/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7000 - loss: 0.6090
Epoch 5/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.6948 - loss: 0.6128
Accuracy: 69.61507797241211
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWa
rning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
276/276 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - accuracy: 0.6314 - loss: 0.6408
Epoch 2/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7067 - loss: 0.6053
Epoch 3/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7000 - loss: 0.6115
Epoch 4/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.6973 - loss: 0.6134
Epoch 5/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7006 - loss: 0.6101
Accuracy: 69.61507797241211
```

In [24]:
```
# 7. Design and implement LSTM model with tensor flow / keras and check accura
import tensorflow as tf
import pandas as pd

df = pd.read_csv("netflix_titles.csv.csv").dropna(subset=["description"])
df["label"] = (df["type"] == "Movie").astype(int)

tokenizer = tf.keras.preprocessing.text.Tokenizer(char_level=True)
tokenizer.fit_on_texts(df["description"])
X = tokenizer.texts_to_sequences(df["description"])

X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=100).reshape(-1,10
y = df["label"].values

# LSTM Model
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(32, input_shape=(100,1)),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy
model.fit(X, y, epochs=5, verbose=1)

print("Accuracy:", model.evaluate(X, y, verbose=0)[1]*100)
```

```
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWa
rning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
276/276 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.6904 - loss: 0.6226
Epoch 2/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.7057 - loss: 0.6066
Epoch 3/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6937 - loss: 0.6169
Epoch 4/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.6927 - loss: 0.6168
Epoch 5/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.6923 - loss: 0.6184
Accuracy: 69.61507797241211
```

In [22]:
```
# ==================== 8. GRU MODEL ====================
import tensorflow as tf
import pandas as pd

# Load + prepare data
df = pd.read_csv("netflix_titles.csv.csv").dropna(subset=["description"])
df["label"] = (df["type"] == "Movie").astype(int)

tokenizer = tf.keras.preprocessing.text.Tokenizer(char_level=True)
tokenizer.fit_on_texts(df["description"])
X = tokenizer.texts_to_sequences(df["description"])

X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=100).reshape(-1,10
y = df["label"].values

# GRU Model
model = tf.keras.Sequential([
    tf.keras.layers.GRU(32, input_shape=(100,1)),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy
model.fit(X, y, epochs=5, verbose=1)

print("Accuracy:", model.evaluate(X, y, verbose=0)[1]*100)
```

```
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWa
rning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
276/276 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.6488 - loss: 0.6335
Epoch 2/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6994 - loss: 0.6133
Epoch 3/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - accuracy: 0.6952 - loss: 0.6162
Epoch 4/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.6888 - loss: 0.6221
Epoch 5/5
276/276 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.6996 - loss: 0.6115
Accuracy: 69.61507797241211
```

In [23]:

```
# ==================== 9. CNN FOR JPG IMAGES (CIFAR-10) ====================
import tensorflow as tf
import numpy as np
import pickle
from sklearn.model_selection import train_test_split

# Load CIFAR-10 batch
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding="bytes")

X = batch[b"data"].reshape(-1, 32, 32, 3) / 255.0
```

```
y = np.array(batch[b"labels"])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# CNN Model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metric
model.fit(X_train, y_train, epochs=10, verbose=0)

print("Accuracy:", model.evaluate(X_test, y_test, verbose=0)[1] * 100)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Accuracy: 52.85000205039978
```

In [49]:
```
# ==================== 10. Minimal CNN for multi-category TIFF images ========

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Drop
from tensorflow.keras.regularizers import l2


data_dir = "/content/Deep-Learning-Lab/CIFAR10_TIFF"
img_size = (32, 32)
batch_size = 32

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_gen = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_gen = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

model = Sequential([
    Conv2D(32, (3,3), activation='relu', kernel_regularizer=l2(0.001), input_s
    MaxPooling2D((2,2)),
    Dropout(0.25),

    Conv2D(64, (3,3), activation='relu', kernel_regularizer=l2(0.001)),
    MaxPooling2D((2,2)),
    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.5),
    Dense(train_gen.num_classes, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=30,
    verbose = 0
)

val_loss, val_acc = model.evaluate(val_gen)
print("Validation Accuracy:", val_acc)
```

```
Found 8003 images belonging to 10 classes.
/usr/local/lib/python3.12/dist-packages/keras/src/legacy/preprocessing/image.p
y:146: UserWarning: Using ".tiff" files with multiple bands will cause distorti
on. Please verify your output.
  warnings.warn(
Found 1997 images belonging to 10 classes.
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dat
aset_adapter.py:121: UserWarning: Your `PyDataset` class should call `supe
r().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`,
`use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fi
t()`, as they will be ignored.
  self._warn_if_super_not_called()
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6294 - loss: 1.3128
Validation Accuracy: 0.6334501504898071
```

In [38]:
```
# ==================== 11. CNN architecture (LeNet) ====================
```

```python
import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, AveragePooling2D, Flatten, Dense,
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)

# Build LeNet model
model = Sequential([
    Conv2D(6, (5,5), activation='relu', input_shape=(32,32,3), padding='same')
    AveragePooling2D(pool_size=(2,2)),
    Conv2D(16, (5,5), activation='relu'),
    AveragePooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(120, activation='relu'),
    Dense(84, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.5177 - loss: 1.4142
Test Accuracy: 0.5070000290870667
```

In [40]:
```python
# ==================== 11. CNN architecture (AlexNet) ====================

import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Drop
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)

# Build AlexNet model
model = Sequential([
    Conv2D(64, (3,3), activation='relu', padding='same', input_shape=(32,32,3)
    MaxPooling2D((2,2)),
    Conv2D(128, (3,3), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Conv2D(256, (3,3), activation='relu', padding='same'),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc
# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.5759 - loss: 1.1964
Test Accuracy: 0.5709999799728394
```

In [41]:
```
# ==================== 11. CNN architecture (ZF-Net) ====================

import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Drop
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)

# Build ZF-Net model
model = Sequential([
    Conv2D(64, (7,7), strides=2, activation='relu', padding='same', input_shap
    MaxPooling2D((2,2)),
    Conv2D(192, (5,5), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Conv2D(384, (3,3), activation='relu', padding='same'),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.5175 - loss: 1.3339
Test Accuracy: 0.5180000066757202
```

In [42]:
```
# ==================== 11. CNN architecture (VGG-11) ====================

import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Drop
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)
```

```python
# Build VGG-11 model
model = Sequential([
    Conv2D(64, (3,3), activation='relu', padding='same', input_shape=(32,32,3)
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(128, (3,3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(256, (3,3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.6130 - loss: 1.0602
Test Accuracy: 0.6115000247955322
```

In [43]:
```python
# ==================== 11. CNN architecture (GoogLeNet) ====================

import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dens
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
```

```
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)

# Define simplified inception block
def inception_block(x, filters):
    f1, f3, f5 = filters
    path1 = Conv2D(f1, (1,1), activation='relu', padding='same')(x)
    path2 = Conv2D(f3, (3,3), activation='relu', padding='same')(x)
    path3 = Conv2D(f5, (5,5), activation='relu', padding='same')(x)
    return Concatenate()([path1, path2, path3])

# Build model
inputs = Input(shape=(32,32,3))
x = inception_block(inputs, [32,32,32])
x = MaxPooling2D(pool_size=(2,2))(x)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
outputs = Dense(10, activation='softmax')(x)

model = Model(inputs, outputs)

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.5395 - loss: 1.2918
Test Accuracy: 0.534500002861023
```

In [44]:

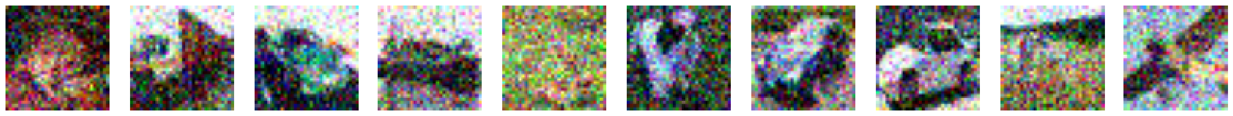
```
# ==================== 11. CNN architecture (ResNet) ====================

import pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activat
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping

# Load data_batch_1
with open("data_batch_1", "rb") as f:
    batch = pickle.load(f, encoding='bytes')
X = batch[b'data']
y = np.array(batch[b'labels'])

# Reshape to images (N, 32, 32, 3)
X_images = X.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1).astype("float32") /

# One-hot encode labels
y_cat = to_categorical(y, num_classes=10)

# Train-Test split
X_train, X_test, y_train, y_test = train_test_split(X_images, y_cat, test_size
print("Data loaded:", X_train.shape, X_test.shape)

# Define simplified residual block
def res_block(x, filters):
    shortcut = x
    x = Conv2D(filters, (3,3), padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters, (3,3), padding='same')(x)
    x = BatchNormalization()(x)
    x = Add()([shortcut, x])
    x = Activation('relu')(x)
    return x

# Build model
inputs = Input(shape=(32,32,3))
x = Conv2D(64, (3,3), padding='same', activation='relu')(inputs)
x = res_block(x, 64)
x = res_block(x, 64)
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
outputs = Dense(10, activation='softmax')(x)

model = Model(inputs, outputs)

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weight

# Train model
history = model.fit(X_train, y_train, epochs=30, batch_size=64, validation_spl

# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test)
```

```
print("Test Accuracy:", test_acc)
```

```
Data loaded: (8000, 32, 32, 3) (2000, 32, 32, 3)
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.4296 - loss: 1.5599
Test Accuracy: 0.43299999833106995
```

In [55]:
```
# ==================== 12. Auto encoder to de-noise image ====================
import pickle, numpy as np, matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, UpSampling2D

with open("data_batch_1", "rb") as f:
    b = pickle.load(f, encoding='bytes')

X = b[b'data'].reshape(-1, 3, 32, 32).transpose(0,2,3,1) / 255.0

# --- Add noise ---
X_noisy = np.clip(X + 0.2*np.random.randn(*X.shape), 0, 1)

X_train, X_test, Y_train, Y_test = train_test_split(X_noisy, X, test_size=0.2,

model = Sequential([
    Conv2D(32, 3, activation='relu', padding='same', input_shape=(32,32,3)),
    MaxPooling2D(2, padding='same'),
    Conv2D(16, 3, activation='relu', padding='same'),
    MaxPooling2D(2, padding='same'),
    UpSampling2D(2),
    Conv2D(32, 3, activation='relu', padding='same'),
    UpSampling2D(2),
    Conv2D(3, 3, activation='sigmoid', padding='same')
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, Y_train, epochs=5, batch_size=64,verbose = 0, validation_da

print("Test MSE:", model.evaluate(X_test, Y_test, verbose=0))

decoded = model.predict(X_test[:10])

plt.figure(figsize=(10,4))
for i in range(10):
    plt.subplot(2,10,i+1); plt.imshow(X_test[i]); plt.axis("off")
    plt.subplot(2,10,i+11); plt.imshow(decoded[i]); plt.axis("off")
plt.show()
```

```
Test MSE: 0.008036181330680847
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 578ms/step
```



```
In [ ]:
```