Building Neural Networks for Image Classification

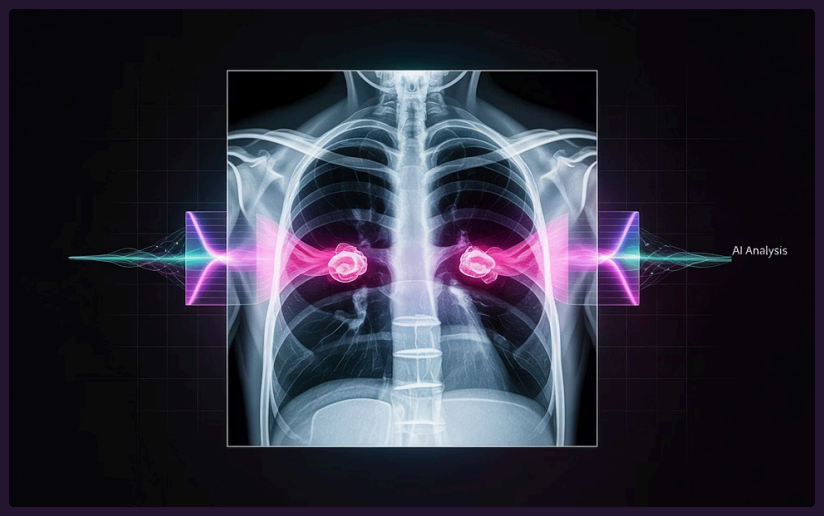
Learn how to harness the power of neural networks for image classification. We'll explore CNN architecture and implement models using TensorFlow.

by S MM

What is Image Classification?

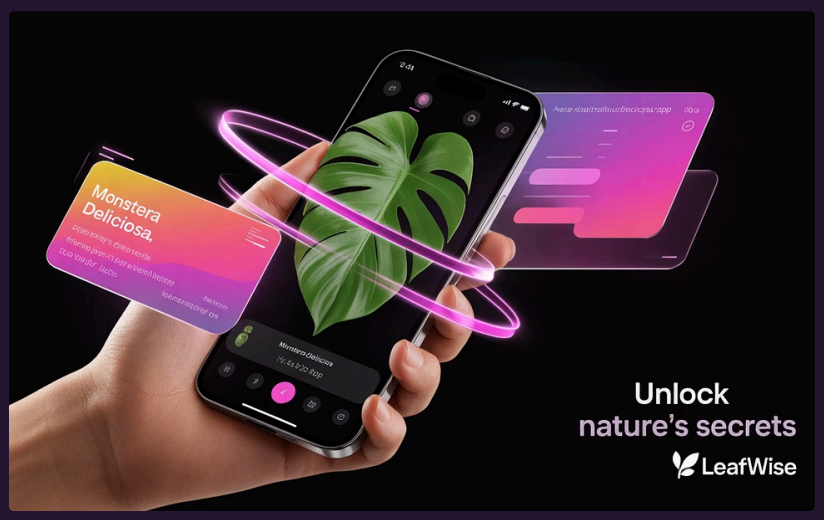
Security Systems

Face recognition software identifies individuals from security footage. Used in airports, stadiums, and businesses.



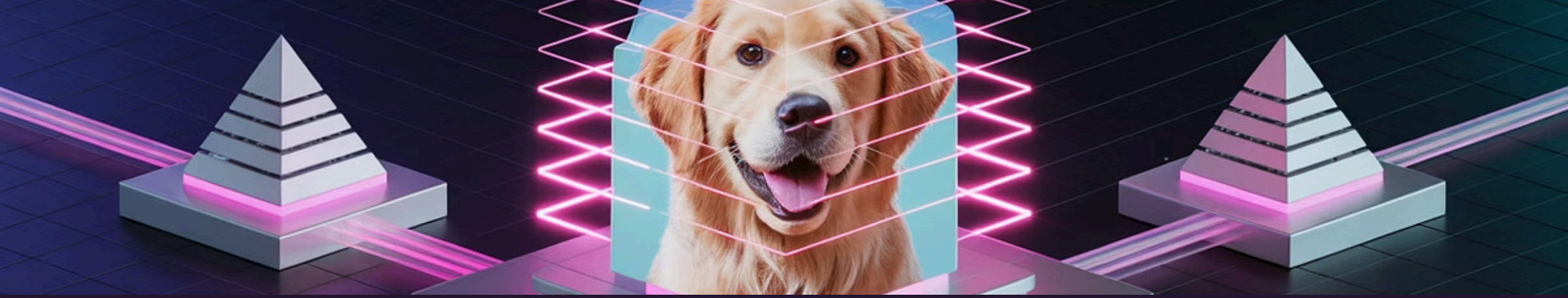
Medical Diagnostics

AI systems detect abnormalities in medical images. Can identify tumors, fractures, and other conditions.



Consumer Applications

Mobile apps identify plants, animals, landmarks, and products. Makes information accessible instantly.



Convolutional Neural Networks (CNNs)

Convolutional Layers

Extract features using filters that scan across the image. Detect edges, textures, and patterns.

Pooling Layers

Reduce dimensions while preserving important information. Makes computation more efficient.

Fully Connected Layers

Combine features for final classification. Connect all neurons between layers.

Popular Image Datasets

MNIST

70,000 handwritten digits (0-9)

28×28 pixel grayscale images

Standard benchmark for beginners

CIFAR-10

60,000 color images in 10 classes

32×32 pixel RGB images

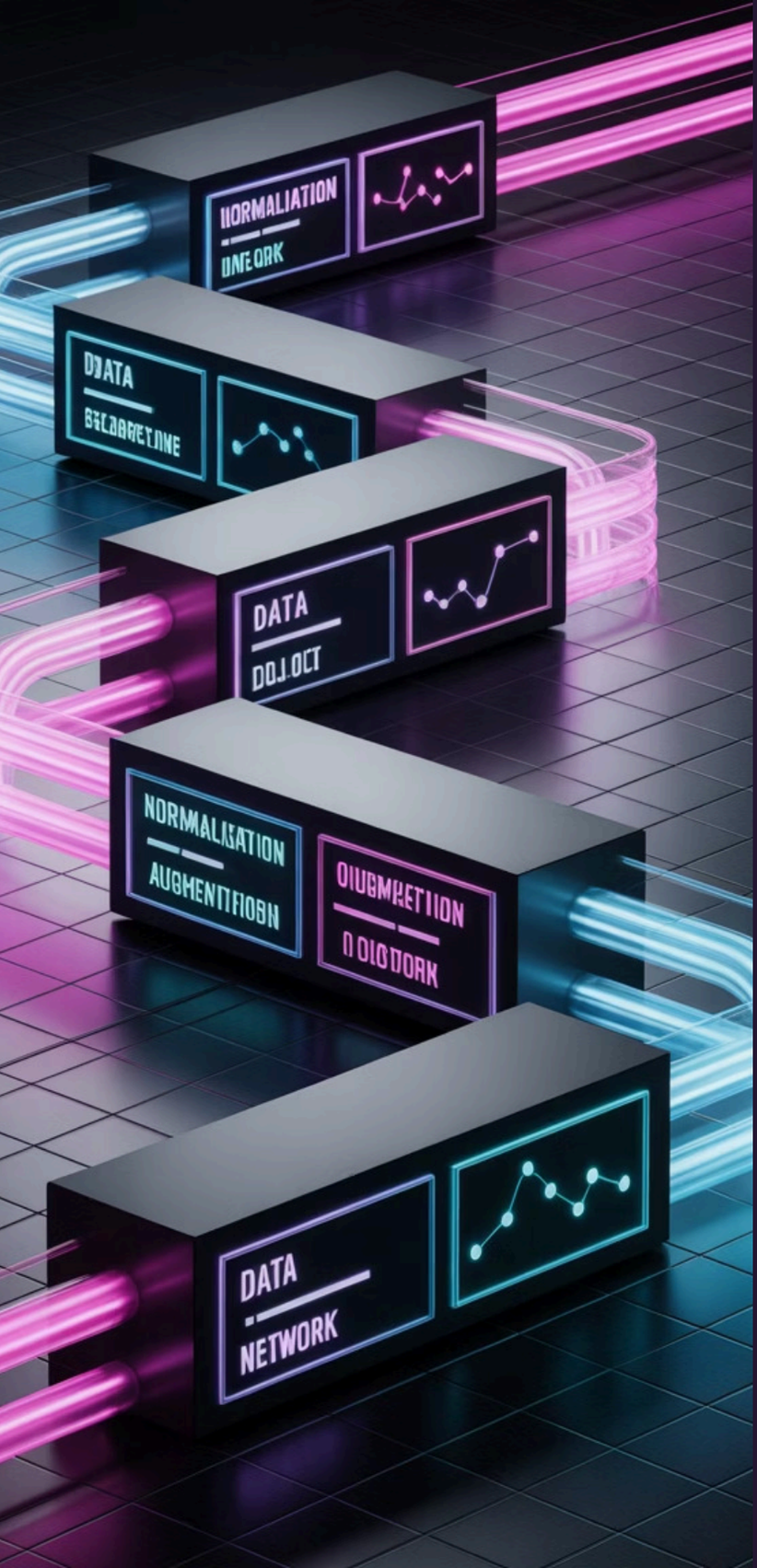
Includes animals, vehicles, and objects

Custom Datasets

Created for specific applications

Requires careful collection and labeling

Can use data augmentation to expand



TensorFlow Workflow: Load & Preprocess

Import Data

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
```

Normalize Pixel Values

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Scale pixel values to range [0,1] for better training performance.

Reshape for CNN Input

```
train_images = train_images.reshape(-1, 28, 28, 1)
test_images = test_images.reshape(-1, 28, 28, 1)
```

TensorFlow Workflow: Define & Train

Define Model

```
model = Sequential([
  Conv2D(32, (3,3), activation='relu',
         input_shape=(28,28,1)),
  MaxPooling2D((2,2)),
  Conv2D(64, (3,3), activation='relu'),
  MaxPooling2D((2,2)),
  Flatten(),
  Dense(64, activation='relu'),
  Dense(10, activation='softmax')
])
```

Compile & Train

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

history = model.fit(
  train_images,
  train_labels,
  epochs=10,
  validation_data=(test_images, test_labels)
)
```

TensorFlow Workflow: Evaluate

1 Test Accuracy

```
test_loss, test_acc =
model.evaluate(test_images,
test_labels)
print(f"Test accuracy:
{test_acc:.4f}")
```

Measures model performance on unseen data. Aim for high accuracy without overfitting.

2 Confusion Matrix

```
from sklearn.metrics import
confusion_matrix
predictions =
model.predict(test_images)
pred_classes =
np.argmax(predictions,
axis=1)
conf_mat =
confusion_matrix(test_labels,
pred_classes)
```

Shows prediction patterns across classes. Helps identify where model struggles.

3 Visualization

Plot accuracy/loss curves to detect overfitting. Compare training and validation metrics.