Minor in AI

TinyML

CNNs & Transfer Learning

May 28, 2025

1. Convolutional Neural Networks (CNNs)

1.1 Architecture Overview

What is a CNN?

A CNN is a type of deep neural network specifically designed to process data with grid-like topology, such as images. It mimics the visual cortex, capturing spatial hierarchies of features through its layers.

- **Input Layer:** Accepts raw image data (e.g., size $28 \times 28 \times 1$ for grayscale).
- **Convolutional Layers:** Apply filters (kernels) to extract features like edges, textures, and shapes.
- **Activation Functions:** Usually ReLU (Rectified Linear Unit) is applied after convolution to introduce non-linearity.
- **Pooling Layers:** Perform downsampling (e.g., MaxPooling) to reduce spatial dimensions and computation.
- **Flattening:** Converts 2D outputs from pooling into a 1D vector before feeding to dense layers.
- **Fully Connected Layers:** Act as a traditional neural network classifier.
- **Output Layer:** Gives final prediction probabilities (via softmax in classification tasks).

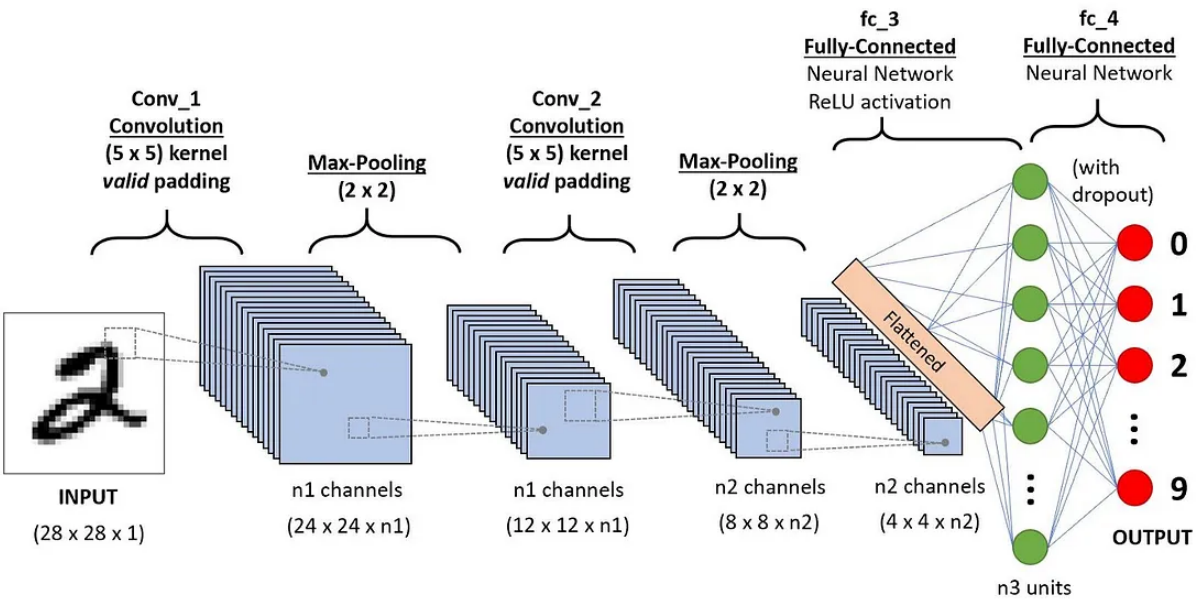

Figure 1: CNN

Try This!

Draw the architecture of a basic CNN used for image classification on Fashion MNIST. Label each layer and specify the input/output dimensions.

2. Transfer Learning

Transfer Learning in Action

Transfer learning involves reusing a pre-trained model (like MobileNet, VGG16, ResNet) and adapting it to a new task. This saves time and resources and boosts performance, especially when data is limited.

Steps to Implement Transfer Learning

1. Load a pre-trained model **without** its top (classifier) layers.
2. Freeze the convolutional base (optional).
3. Add new **custom dense layers** for your task.
4. Train the new layers on your dataset (e.g., Fashion MNIST).

Reflect

Why do we often freeze the base model when doing transfer learning? What might happen if we don't?

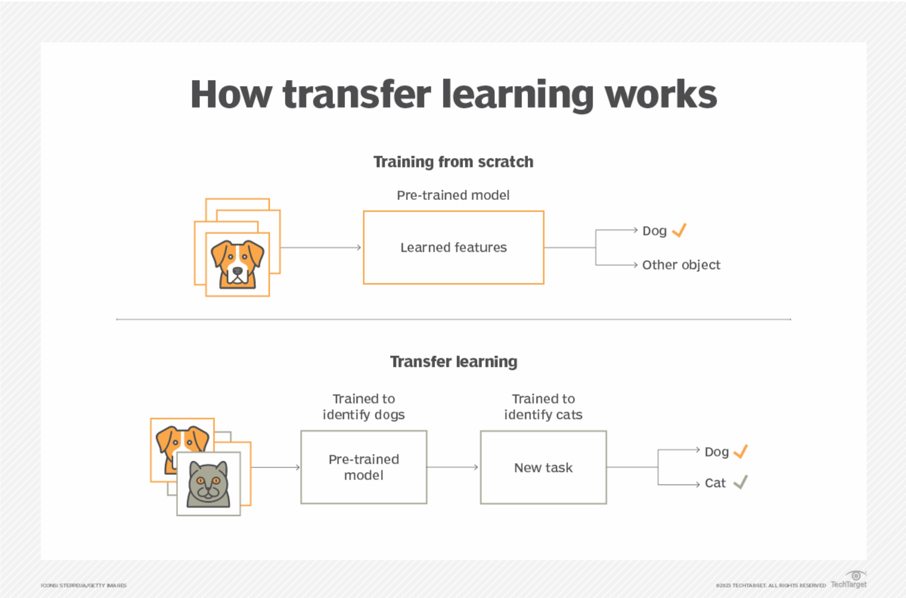

Figure 2: Tranfer Learning

3. Model Optimization Techniques

3.1 Quantization

- Converts model weights from `float32` to `float16` or `int8`.
- **Benefit:** Reduced memory footprint, faster inference.
- Useful in edge devices or mobile deployment.

Code Snippet (TensorFlow Lite Quantization)

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
quantized_model = converter.convert()
```

3.2 Pruning

- Removes low-weight connections (neurons or edges) during training.
- Leads to sparser and smaller models.
- Often used with regularization to avoid performance degradation.

Think & Write

Compare pruning and dropout. How are they similar? How are they different in terms of purpose and execution?

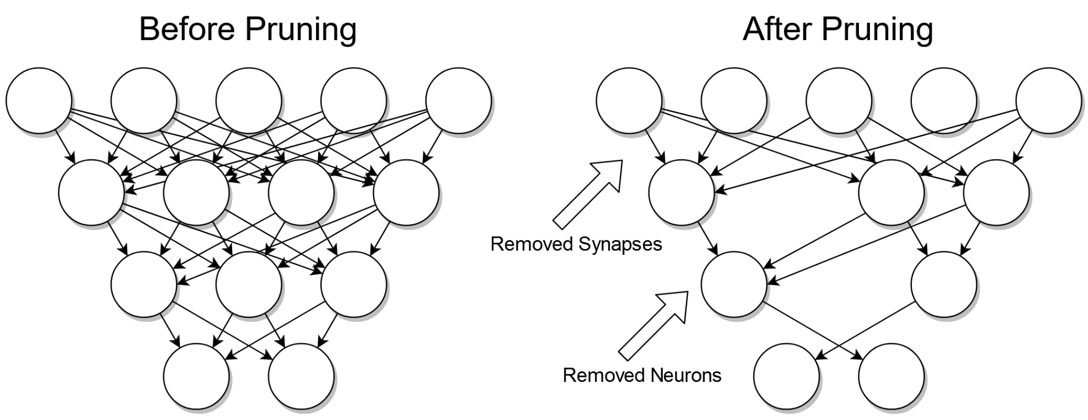

Figure 3: Pruning

4. Case Study: Human Activity Classification

4.1 Model Used: MobileNet V2

- Lightweight and efficient architecture.
- Combines depthwise separable convolutions with residual connections.
- Ideal for mobile and embedded devices.

4.2 Implementation Overview

- Preprocessing sensor data into time-series or spectrogram formats.
- Using MobileNetV2 as a feature extractor.
- Training a custom classifier on top.

Model Performance Insight

Compare the model size and accuracy before and after quantization and pruning. How does performance trade off against model size?

5. Key Takeaways

1. CNNs extract hierarchical features from images using convolution and pooling.
2. Transfer learning allows reuse of powerful pre-trained models.
3. Optimization techniques like quantization and pruning make models smaller and faster for deployment.
4. MobileNetV2 is an efficient backbone for on-device inference.