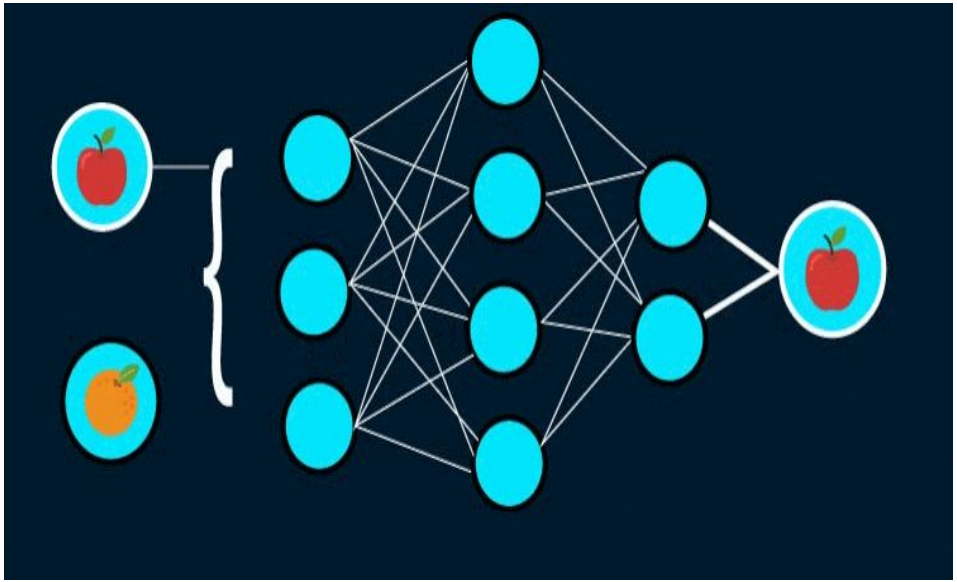
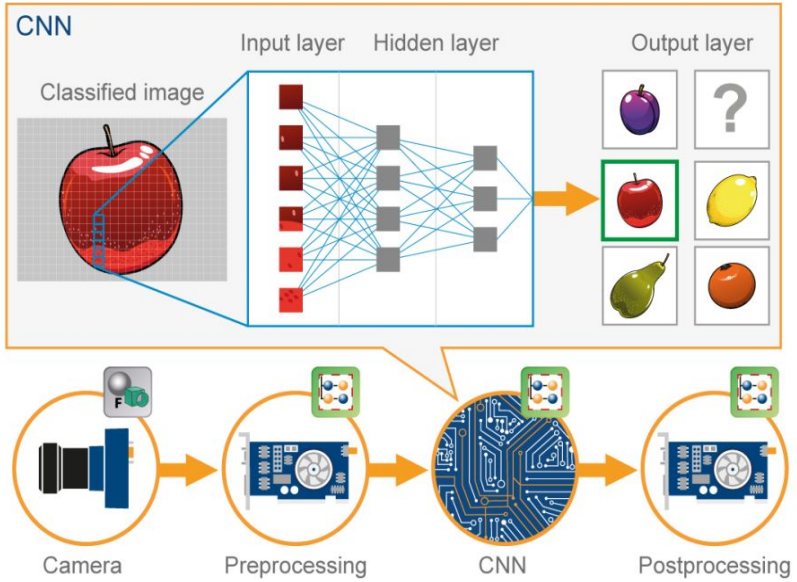
Building Lightweight Architectures

Contents

- Convolutional Neural Networks (CNN)
- Stages of CNN
- Model Building
- Lightweight Models
- Comparison of Model Variants
- Case Study

Convolutional Neural Networks (CNN)



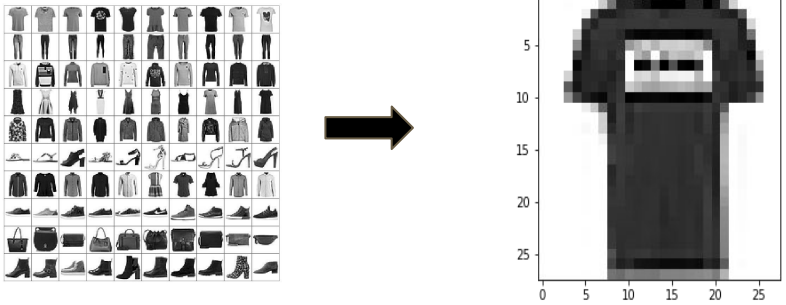


Reference :

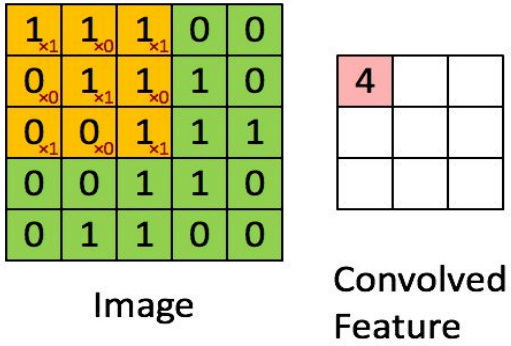
https://www.analyticsvidhya.com/blog/2021/06/image-classification-using-convolutional-neural-network-with-python/

Stages of CNN

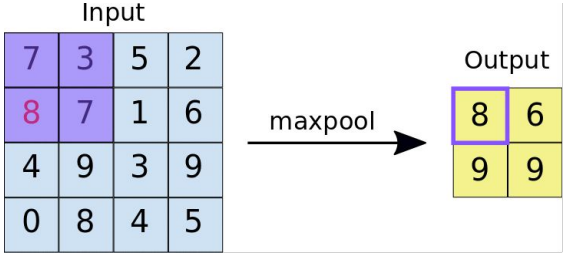
❖ Input Layer



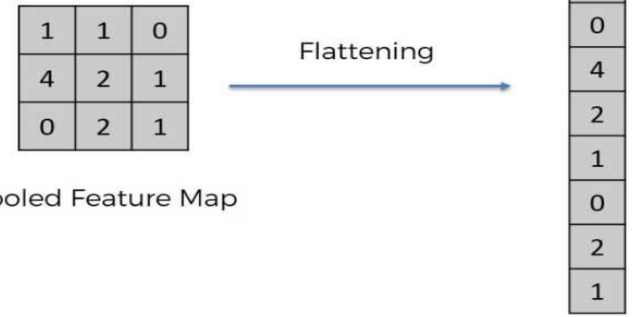
❖ Convolution



❖ Pooling

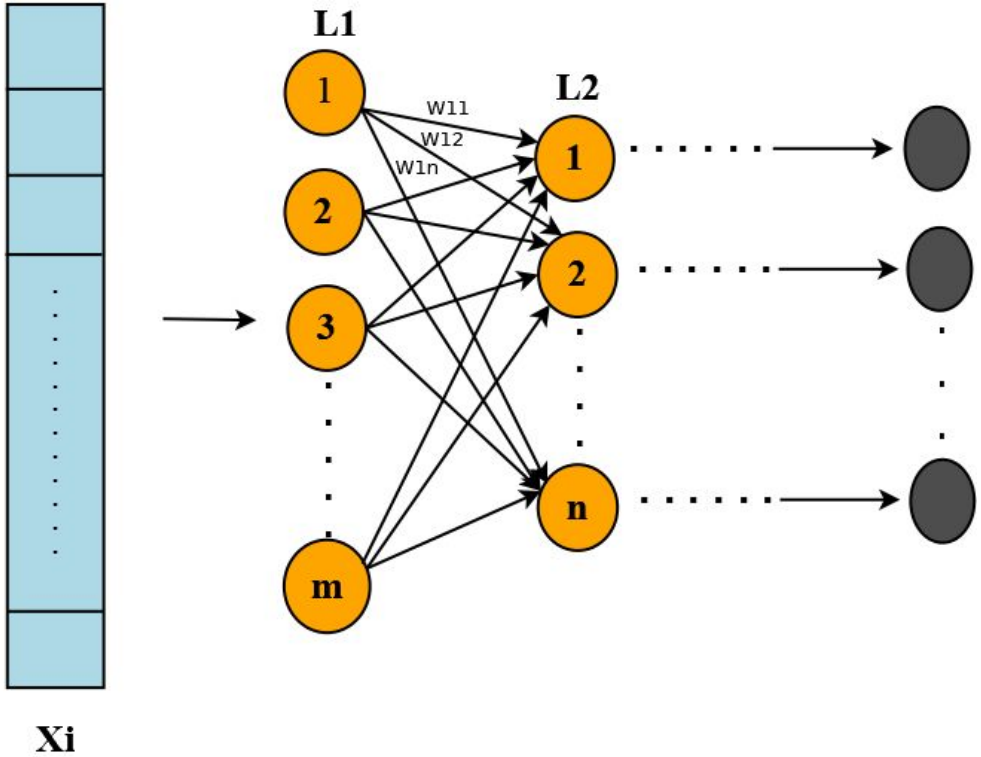


❖ Flattening



Stages of CNN

- Neural Network



Model Building (Transfer Learning)

- Select a Pretrained Model
- Remove the Top (Classification) Layer
- Adding Own Classifier
- Freeze Layers (optional)
- Fine-tune the Model

Domain	Transfer from	Use for
Image Classification	ImageNet-trained CNN	Medical images, fashion, IoT
NLP	BERT, GPT, RoBERTa	Text classification, QA
Audio	VGGish, YAMNet	Emotion detection, sound events
Edge AI	MobileNet/TinyML models	Deploying on microcontrollers

Model Building (MNIST-MobileNetV2)

- **Loading data**

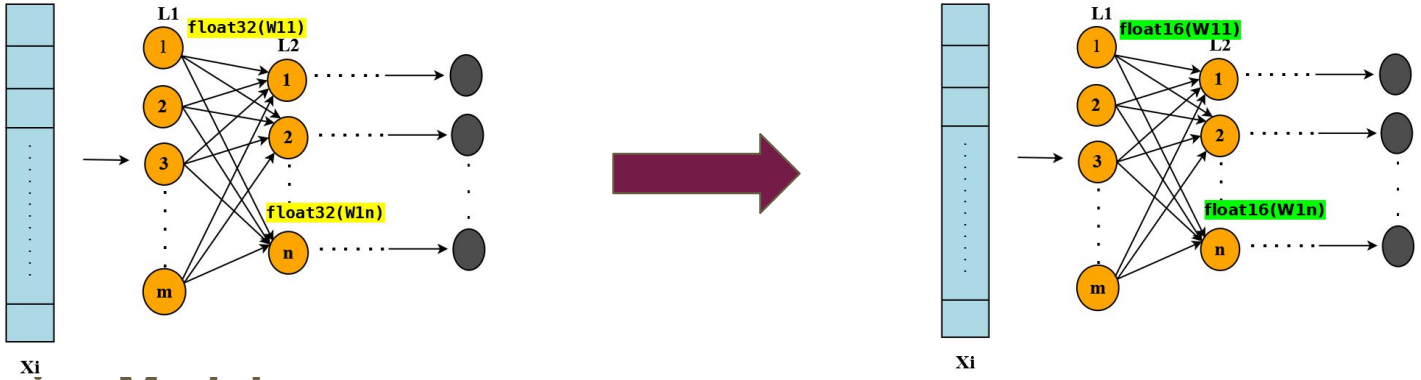
```
# Load and preprocess MNIST
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

train_images = np.stack([train_images] * 3, axis=-1)  # Convert to 3-channel
test_images = np.stack([test_images] * 3, axis=-1)
train_images = tf.image.resize(train_images, [96, 96]) / 255.0
test_images = tf.image.resize(test_images, [96, 96]) / 255.0
```

- **Building Model**

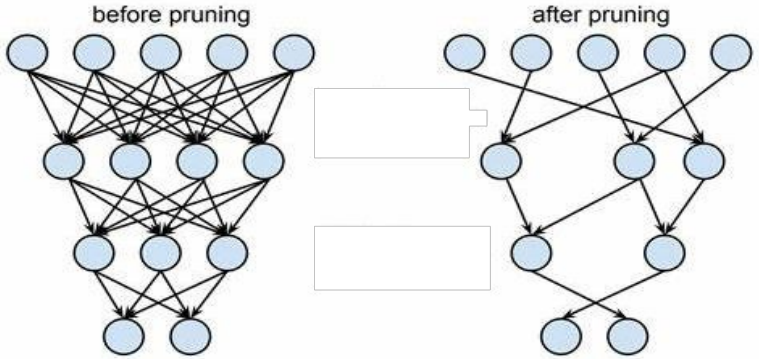
```
# ------------------ BASE MODEL ------------------
def build_transfer_model():
    base_model = tf.keras.applications.MobileNetV2(input_shape=(96, 96, 3),
                                                    include_top=False,
                                                    weights='imagenet')
    base_model.trainable = False

    model = tf.keras.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(128, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])
    return model

base_model = build_transfer_model()
```

Lightweight Models

- Quantization of Models



- Pruning Models



Comparison of Model Variants

Model Version	Using MobileNet		Using SqueezeNet	
	Model Size	Model Accuracy (%)	Model Size	Model Accuracy (%)
Base Model	10.81 MB	89.64	0.82 MB	88.94%
Quantized Model	4.58 MB	81.02	0.13 MB	91.59%
Pruned Model	9.54 MB	80.96	0.30 MB	91.58%

Case Study

- **Develop a deep learning model to classify human activities (e.g., walking, running, sitting) using accelerometer and gyroscope data — and deploy it on a low-power microcontroller such as the Arduino Nano 33 BLE Sense.**

Thank you