Book Report

CNN, BERT-CNN, QCNN, EQCNN

Rifah Sajida Deya

9th January, 2025

Definitions & Examples:

1. CNN (Convolutional Neural Network):

   A Convolutional Neural Network (CNN) is a type of deep learning model commonly used for analyzing visual data, but it can also be applied to other domains like text and speech.

   Key features include:

   1. Convolution Layers: Detect patterns or features (e.g., edges in images or n-grams in text) using filters.
   2. Pooling Layers: Reduce the spatial dimensions of the data, retaining essential features while reducing computational cost.
   3. Fully Connected Layers: Combine features learned in previous layers for classification or regression tasks.
   4. Applications: Image classification, object detection, and even text classification.

   Example: Handwritten digit classification

   Models: LeNet, AlexNet, ResNet, etc.

Padding

Definition: Padding means adding extra pixels (usually zeros) around the edges of an image before applying convolution.

It helps to:

- Keep the image size the same after convolution.
- Prevent shrinking of the image as layers go deeper.
- Allow edge features (like borders) to be learned properly.

Difference between Dense Layer, Pooling Layer, and Convolution Layer:

Layer	Meaning	Purpose
Dense Layer	A normal fully connected layer where every neuron connects to every input.	Used for **final decision-making** (like classification).
Pooling Layer	Reduces the size of the feature map by taking the maximum or average value.	Used to **make the network smaller and faster** and to **focus on important features**.
Convolution Layer	Applies filters (small matrices) to the input to extract important patterns like edges and shapes.	Used for **feature extraction** from the input (especially images).

In short:

- Dense Layer = Connects everything together.
- Pooling Layer = Shrinks and highlights important parts.
- Convolution Layer = Detects patterns in small areas.

Why use ConvNets (Convolutional Neural Networks) over Feed-Forward Neural Networks?

=> **Feed-Forward Neural Networks (FNNs)** treat all input features equally and separately, so they don't understand the **structure** of images (like nearby pixels).

- **ConvNets (CNNs)** are **designed for images** — they can **capture patterns** like edges, textures, and shapes.
- CNNs use fewer parameters (weights) and **focus on important features** using small filters, making them **faster and better** for image tasks.

In short:

> **ConvNets are better for images** because they understand patterns and use fewer resources than normal feed-forward networks.

Example:

- Feed-Forward Network might see an image as just 1000 numbers.
- ConvNet sees small regions and detects features like eyes, nose, etc.

2. BERT-CNN:

   BERT-CNN combines the power of BERT (Bidirectional Encoder Representations from Transformers) for contextual understanding of text with CNN for feature extraction. It is used in natural language processing (NLP) tasks where contextual representation and local feature extraction are both important.

   - BERT's Role: Generates contextual embeddings of text, understanding the meaning of words in relation to their context.

- CNN's Role: Extracts local patterns or features (e.g., n-grams or phrases) from BERT embeddings and processes them for tasks like classification.
- Advantages:
    - BERT provides a rich understanding of language.
    - CNN enhances the model's ability to focus on relevant parts of the text.
    - Applications: Sentiment analysis, text classification, and other NLP tasks.

Example: Sentiment analysis on text

Models/Tools: BERT + CNN, Hugging Face Transformers, etc.

3. QCNN (Quantized Convolutional Neural Network):

QCNN applies quantization techniques to a standard CNN to reduce its size and computational requirements, making it suitable for resource-constrained environments (e.g., edge devices, mobile phones).

- Quantization: Converts the weights and activations of a CNN from floating-point precision to lower precision (e.g., 8-bit integers).
- Advantages:
    - Reduces memory usage.
    - Increases inference speed.
    - Minimizes power consumption.
- Challenges: Quantization can lead to a slight drop in model accuracy.

Example: Image classification on mobile

Models/Tools: TensorFlow Lite, PyTorch Quantization, etc.

4. EQCNN (Efficient Quantized Convolutional Neural Network):

EQCNN extends the concept of QCNN with additional optimizations for better performance, typically focused on maintaining a balance between efficiency and accuracy.

- Optimizations:
  - Advanced quantization methods (e.g., mixed-precision quantization).
  - Model compression techniques like pruning.
  - Efficient architectural designs.
- Applications: Ideal for deploying deep learning models on low-power devices like IoT sensors or smartphones.
- Key Feature: It ensures minimal accuracy degradation despite aggressive compression.

Example: Real-time object detection on IoT devices

Models/Tools: YOLO, ONNX Runtime for Edge AI, etc.

Summary Table:

Model	Purpose	Key Features	Applications
CNN	Feature extraction for visual or sequential data.	Convolutional filters, pooling, and fully connected layers.	Image analysis, text processing.
BERT-CNN	Combines contextual embeddings with feature extraction.	BERT for language understanding, CNN for localized feature extraction.	NLP tasks like sentiment analysis.
QCNN	Resource-efficient CNN.	Quantized weights and activations for reduced size and computation.	Edge AI, mobile deployment.
EQCNN	Efficient and optimized QCNN.	Advanced quantization, pruning, and efficient architecture for balancing efficiency and accuracy.	IoT devices, low-power systems.

Conclusion

To know more visit:

- https://github.com/rifah07/Introduction_of_Machine_Learning
- https://github.com/rifah07/CNN_Only