Neural Network Architectures for Sequence Data

Types of neural network architectures:

- RNNs:
  - Process sequence data sequentially (one position at a time).
  - Capture sequence order and temporal dependencies but are difficult to parallelize.
- CNNs:
  - Process inputs in parallel using convolution filters.
  - Learn local patterns through a local receptive field, giving CNNs a strong local inductive bias.
- Transformers:
  - Relation to RNNs:

    Like RNNs, Transformers were originally developed for sequence data, such as text (a sequence of tokens) and DNA (a sequence of nucleotides). The input is represented as a sequence of feature vectors (e.g., word embeddings or nucleotide encodings), typically forming a 2D tensor with shape (sequence length$\times$feature dimension).

    Unlike RNNs, Transformers process all positions simultaneously using self-attention rather than sequential recurrence.

    Transformers are also not limited to sequence data and can be applied to images, which are naturally represented as 3D tensors (height$\times$width$\times$channels). In Vision Transformers (ViTs), images are divided into patches, each patch is converted into a feature vector (patch embedding), and the resulting sequence of patch embeddings is processed by the Transformer.
  - Relation to CNNs:

    Like CNNs, Transformers process inputs in parallel. However, CNNs use local convolution with a local receptive field to learn local patterns, whereas Transformers use self-attention, allowing every position to interact with every other position and capture global relationships.

Comparison of neural network architectures:

Architecture	Processing	Main mechanism	Main feature	DNA sequence modeling
RNN	Sequential	Recurrent connections	Sequence order	Sequential processing
CNN	Parallel	Local convolution	Local patterns	Local motif detection
Transformer	Parallel	Self-attention	Global relationships	Long-range regulatory interactions

Applications of neural network architectures across different data types:

Data	RNN	CNN	Transformer
Text	Sequential modeling	Conv1D: local patterns	Self-attention
DNA sequence	Sequential modeling	Conv1D: local motifs	Self-attention: long-range dependencies
Audio / time series	Sequential modeling	Conv1D: local temporal patterns	Self-attention: long-range relationships
Image	$\times$ (typically)	Conv2D: spatial patterns	ViT: patch self-attention

Sequence Data in CNNs

Sequence data are traditionally modeled using RNNs, but they can also be processed using CNNs. Unlike images, which are represented as 3D tensors ((height $\times$ width $\times$ channels), sequence data are represented as 2D tensors (sequence length $\times$

features/channels). Conv1D layers can slide along the sequence axis to learn local patterns.

For example, a DNA sequence can be represented as a 2D tensor with the shape (sequence length$\times 4$nucleotide channels), such as (524288×4), where the four channels represent A, C, G, and T. A Conv1D layer with 512 filters contains 512 learnable kernels. Each kernel has the shape (kernel size $\times$ input channels), such as (15×4). Each kernel learns a different local sequence pattern, and together they produce 512 output feature maps. With stride $= 1$ and "same" padding, each feature map has the shape (524288×1), and the final output is obtained by stacking these feature maps along the channel dimension, resulting in an output shape of (524288×512).

Conv1D and Conv2D

Conv1D and Conv2D differ in the number of dimensions along which the kernel slides: Conv1D slides along one dimension, while Conv2D slides along two dimensions. In both cases, the kernel depth matches the number of input channels, and the number of filters determines the number of output feature channels. Each filter generates one feature map, and all feature maps are stacked along the channel dimension to form the final output. Because they operate along different numbers of sliding dimensions, Conv1D and Conv2D have different kernel shapes and output tensor shapes.

- Conv1D:
  - The kernel slides along one ordered dimension (sequence length).
  - The kernel is a 2D tensor with the shape: (kernel size $\times$ input channels).
  - E.g., DNA sequences use Conv1D because they have one sequence axis and multiple input channels (A, C, G, T).
- Conv2D:
  - The kernel slides along two spatial dimensions (height and width).
  - The kernel is a 3D tensor with the shape: (kernel height $\times$ kernel width $\times$ input channels).
  - E.g., Images use Conv2D because they have two spatial dimensions (height and width), regardless of whether they are grayscale or color.

Encoder–Decoder

The encoder–decoder structure divides a complex neural network into two conceptual modules: an encoder that extracts feature representations from the input and a decoder that transforms these representations into the desired outputs. The primary connection between the two modules is that the encoder output serves as the input to the decoder. This modular design simplifies architecture understanding, design, interpretation, and training.

In U-Net architectures, additional skip connections are introduced between intermediate encoder layers and corresponding decoder layers. These connections preserve fine-grained information that may be lost during encoding, while maintaining the overall encoder–decoder framework.