Recitation 10
Structured Neural Networks

Haoxu Huang

NYU CDS

Apr. 16, 2026

Structured Neural Networks

Overview

Historically, people proposed different neural networks architectures for learning from different data.

In this lab, we will review some most representative architectures mentioned in the lecture to understand

- how they are defined
- the motivation of designing them
- how to implement and train some of them with PyTorch

Structured Neural Networks

Overview

Why do we design different model architectures from FNNs to CNNs, RNNs, and Transformers?

The Concept of Inductive Bias: an architecture combined with a vague prior over its weights (e.g., a standard Gaussian distribution) induces a highly structured prior over functions, $p(f(x; W))$.

Structured Neural Networks

Feedforward Neural Networks

Definition

The basic linear transformation followed by a non-linearity

$$y = \sigma(Wx + b)$$

input $x \in \mathbb{R}^{d_{in} \times 1}$, Weights $W \in \mathbb{R}^{d_{out} \times d_{in}}$, Bias $b \in \mathbb{R}^{d_{out} \times 1}$, Output $y \in \mathbb{R}^{d_{out} \times 1}$, non-linearity activation function σ, (e.g. $\sigma(x) = \frac{1}{1+e^{-x}}$ for sigmoid, $\sigma(x) = \max(0, x)$ for ReLU).

Structured Neural Networks

Feedforward Neural Networks

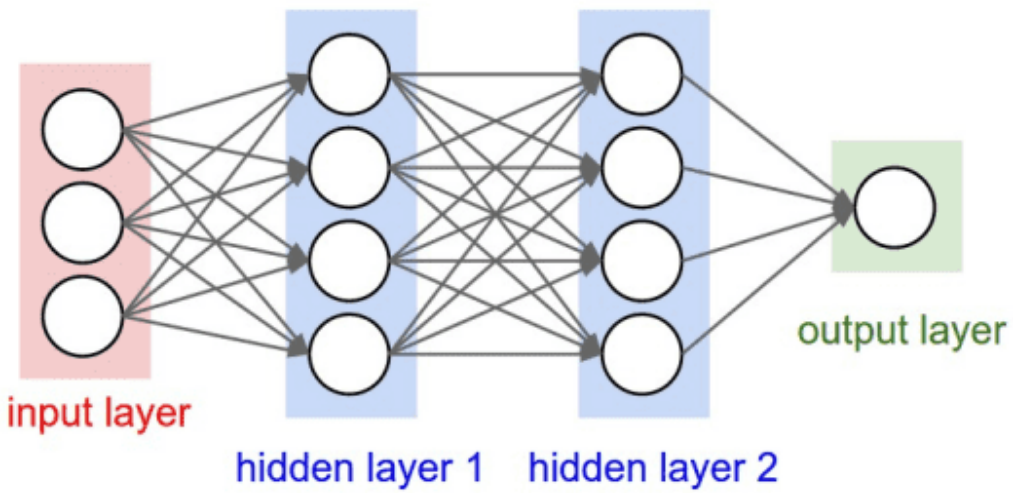
Stacking Layers for Feedforward Neural Networks

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

with

$$a^{(l)} = \sigma(z^{(l)})$$

(l denote the layer index).



Structured Neural Networks

Feedforward Neural Networks

Optimization Binary Cross-Entropy Loss for a batch of m examples.

$$J(W, b) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

for Ground Truth $y_i \in \mathbb{R}$ (scalar values 0 or 1) and Predictions $\hat{y}_i \in \mathbb{R}$ (scalar probabilities).

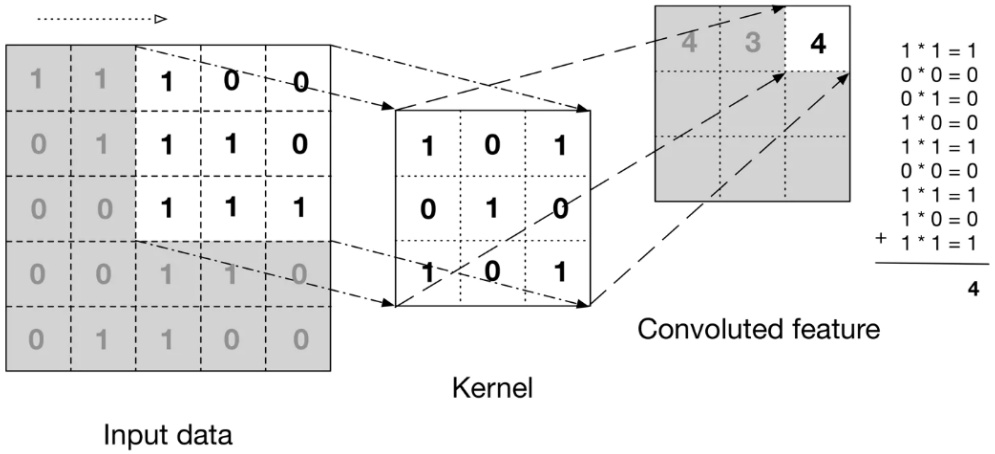
Structured Neural Networks

Convolution Neural Networks

Definition

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

for Input $I \in \mathbb{R}^{H_{in} \times W_{in}}$, Kernel $K \in \mathbb{R}^{k \times k}$, Output Map $S \in \mathbb{R}^{H_{out} \times W_{out}}$ (where $H_{out} = H_{in} - k + 1$). This operation is considered as extracting feature maps.



Structured Neural Networks

Convolution Neural Networks

Downsampling representations for a multi-channel feature map. e.g. Max pooling over region R_{ij}

$$y_{i,j,c} = \max_{(m,n)\in R_{ij}} x_{m,n,c}$$

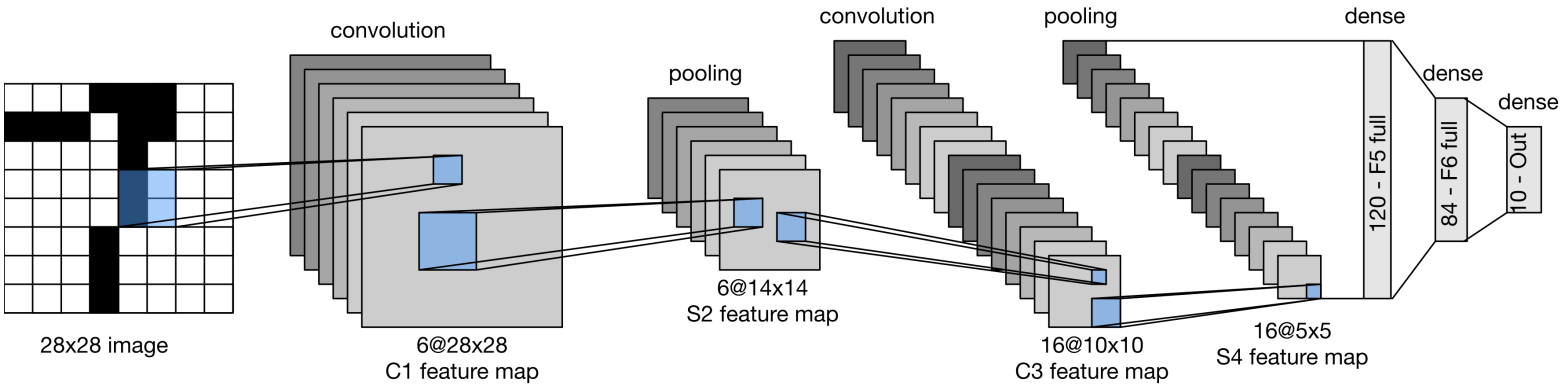
with Input $x \in \mathbb{R}^{H_{in}\times W_{in}\times C}$, Output $y \in \mathbb{R}^{H_{out}\times W_{out}\times C}$ (Channels C remain constant).

The standard pipeline for CNN:
Conv $\rightarrow$ ReLU $\rightarrow$ Pool $\rightarrow$ Flatten $\rightarrow$ Dense.

Structured Neural Networks

Convolution Neural Networks

Full CNN example (LeNet)



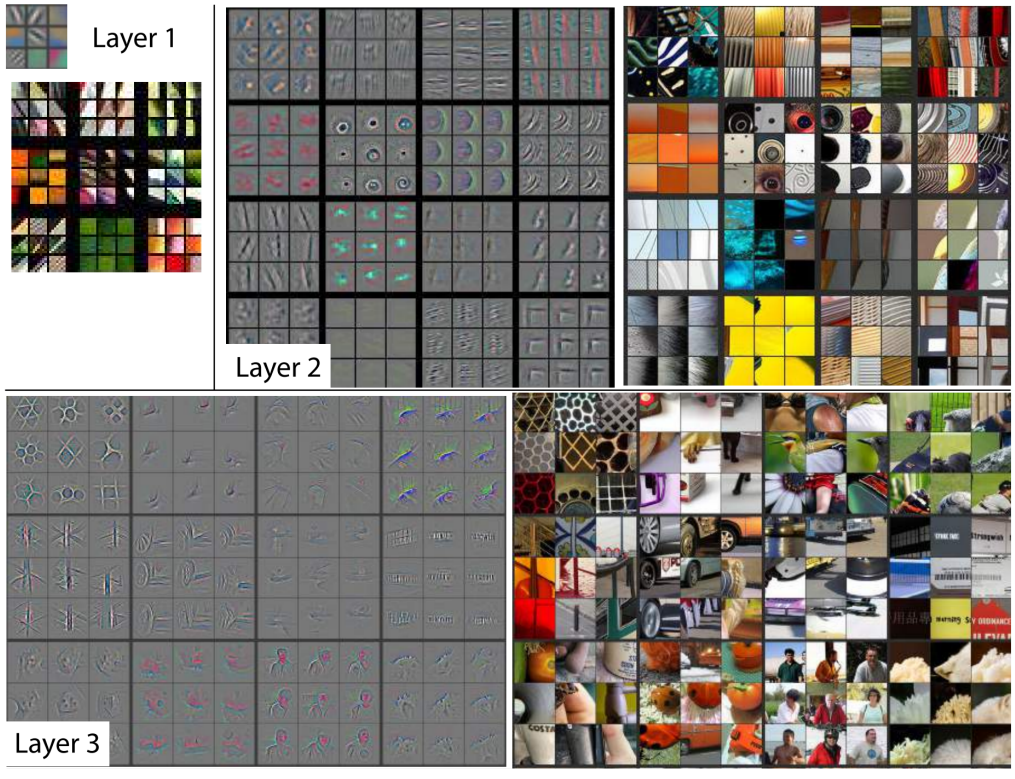
Yann LeCun Leon Bottou Yoshua Bengio and Patrick Haffner. Gradient Based Learning Applied to Document Recognition. Proc of the IEEE 1998.

Structured Neural Networks

Convolution Neural Networks

What feature does each layer of CNN extracting?

- Hierarchical progression where layers move from learning simple features to complex, class-specific objects



Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014.

Structured Neural Networks

Recurrent Neural Networks

Processing temporal sequences $x_1, x_2, ..., x_T$ with recurrence mapping over time step t.

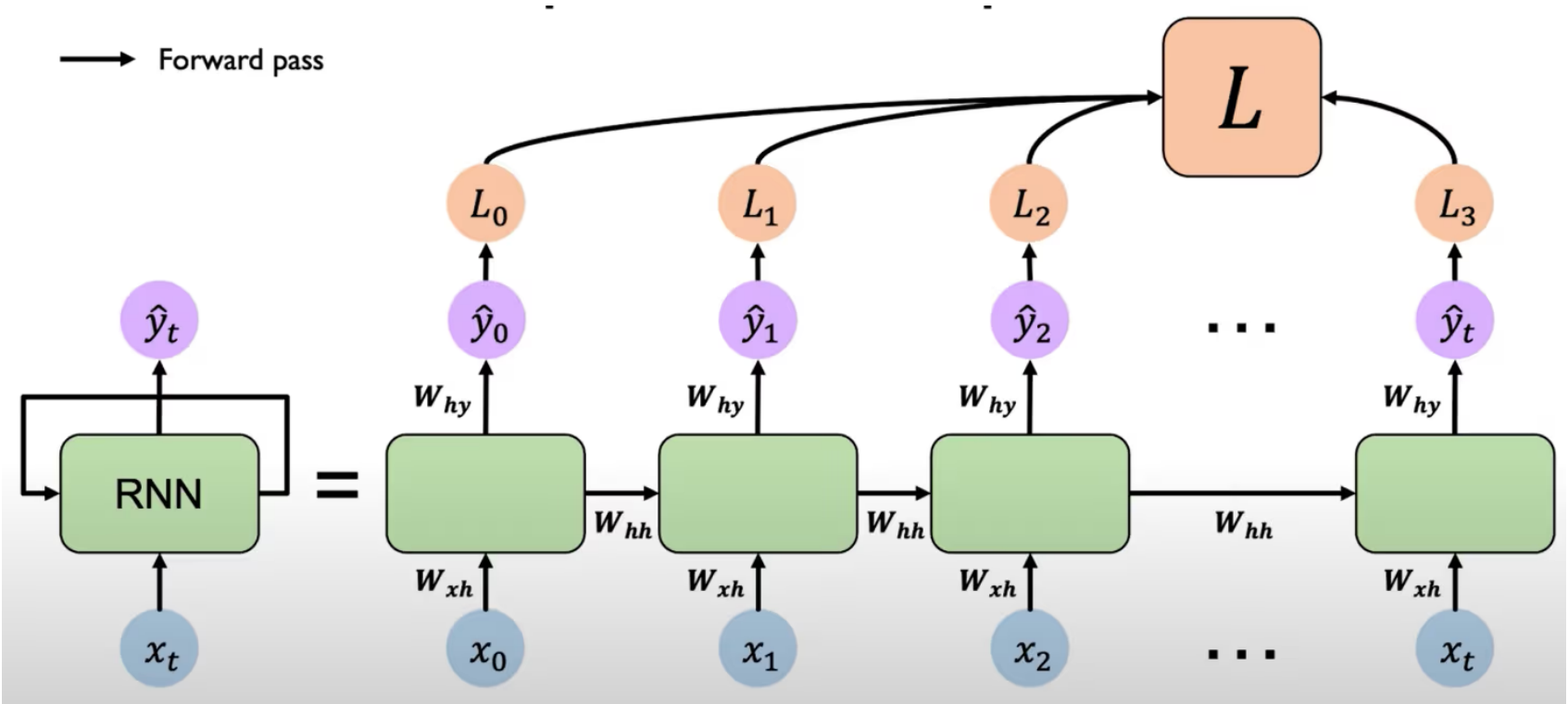
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

$$y_t = W_{hy} h_t + b_y$$

Dimensions: Let input size be d_x, hidden size d_h, output size d_y. $x_t \in \mathbb{R}^{d_x \times 1}, h_t, h_{t-1} \in \mathbb{R}^{d_h \times 1}, W_{hh} \in \mathbb{R}^{d_h \times d_h}, W_{xh} \in \mathbb{R}^{d_h \times d_x}, b_h \in \mathbb{R}^{d_h \times 1}, W_{hy} \in \mathbb{R}^{d_y \times d_h}, b_y \in \mathbb{R}^{d_y \times 1}, y_t \in \mathbb{R}^{d_y \times 1}$

Structured Neural Networks

Recurrent Neural Networks



Structured Neural Networks

Vanishing Gradient Problem

$$\text{Gradient} \propto \prod_{k=1}^{K} (\text{Weight Matrix}) \times (\text{Derivative of Activation})$$

CNN:

$$\frac{\partial x_L}{\partial x_0} = \prod_{l=1}^{L} \frac{\partial x_l}{\partial x_{l-1}} = \prod_{l=1}^{L} W_l \sigma'(z_l)$$

RNN:

$$\frac{\partial h_T}{\partial h_0} = \prod_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} = \prod_{t=1}^{T} W_{hh} \sigma'(z_t)$$

As depth L increases or sequence length T increases, the gradient vanishes. If the weights are initialized to small values (< 1) and you use activation functions like Sigmoid or Tanh (whose maximum derivatives are 0.25 and 1.0 respectively), each term in that product is a fraction.

Structured Neural Networks

Skip Connection

Formulation:

$$x_l = \mathcal{F}(x_{l-1}) + x_{l-1}$$

Gradient:

$$\frac{\partial x_l}{\partial x_{l-1}} = \frac{\partial}{\partial x_{l-1}} \left[\mathcal{F}(x_{l-1}) + x_{l-1}\right] = \frac{\partial \mathcal{F}}{\partial x_{l-1}} + \mathbf{I}$$

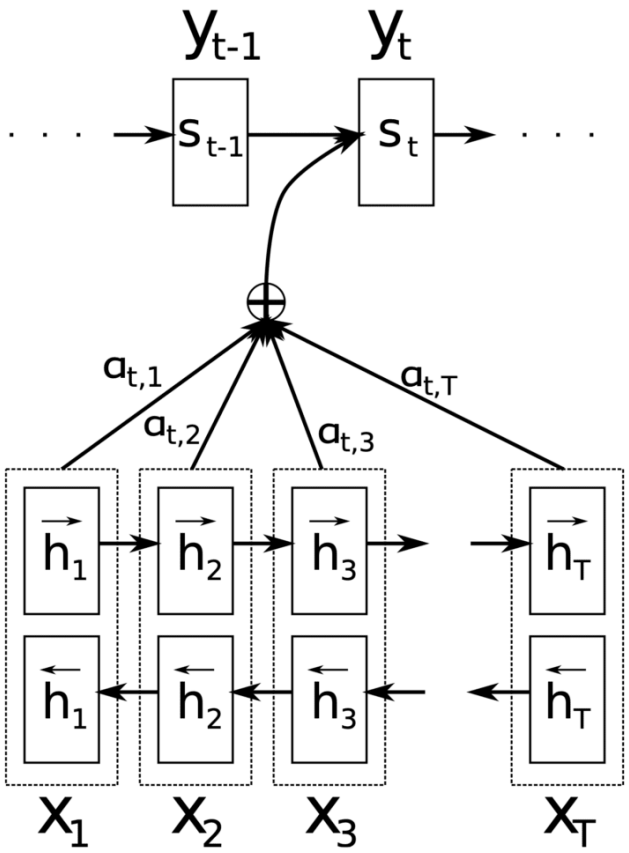
where $\mathbf{I}$ is the Identity matrix.
Then, if we take this derivative back to the chain rule product.

For example, $0.1 \times 0.1 \times 0.1 = 0.001$ (Vanishing) $\rightarrow$
$(0.1 + 1) \times (0.1 + 1) \times (0.1 + 1) = 1.331$ (Preserved)
(This is an intuitive example. In reality, the calculation would be on Jacobian and Identity Matrices)

Structured Neural Networks

Attention/Self-Attention

Allowing the decoder to "look back" at all encoder states.



Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation By Jointly Learning to Align and Translate. ICLR 2015.

Structured Neural Networks

Attention/Self-Attention

Standard Attention Mechanism:

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$

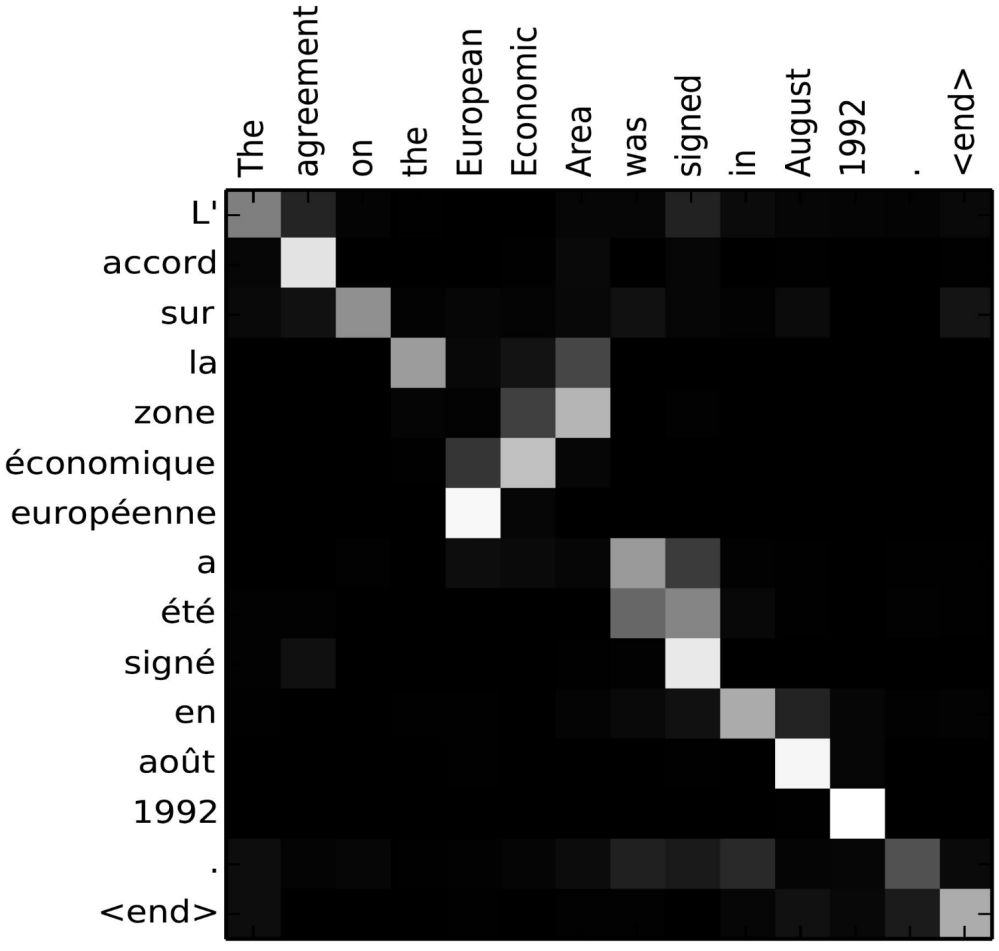
$$e_{tj} = a(s_{t-1}, h_i)$$

Dimensions: Hidden states $h_j \in \mathbb{R}^{d_h \times 1}$, Attention weights $\alpha_{tj} \in \mathbb{R}$ (scalars summing to 1), Context vector $c_t \in \mathbb{R}^{d_h \times 1}$

Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation By Jointly Learning to Align and Translate. ICLR 2015.

Structured Neural Networks

Attention/Self-Attention



Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation By Jointly Learning to Align and Translate. ICLR 2015.

Structured Neural Networks

Self-Attention

What if a sequence attends to itself? Introducing Q, K, and V matrices.

Linear projections of the input X:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dimensions: Input Matrix $X \in \mathbb{R}^{N \times d_{model}}$, Weights $W^Q, W^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W^V \in \mathbb{R}^{d_{model} \times d_v}$, Queries $Q \in \mathbb{R}^{N \times d_k}$, Keys $K \in \mathbb{R}^{N \times d_k}$, Values $V \in \mathbb{R}^{N \times d_v}$, $QK^T \in \mathbb{R}^{N \times N}$, $\text{softmax}(\dots)V \in \mathbb{R}^{N \times d_v}$

Structured Neural Networks

Multihead-Attention

Parallel attention under subspaces.

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

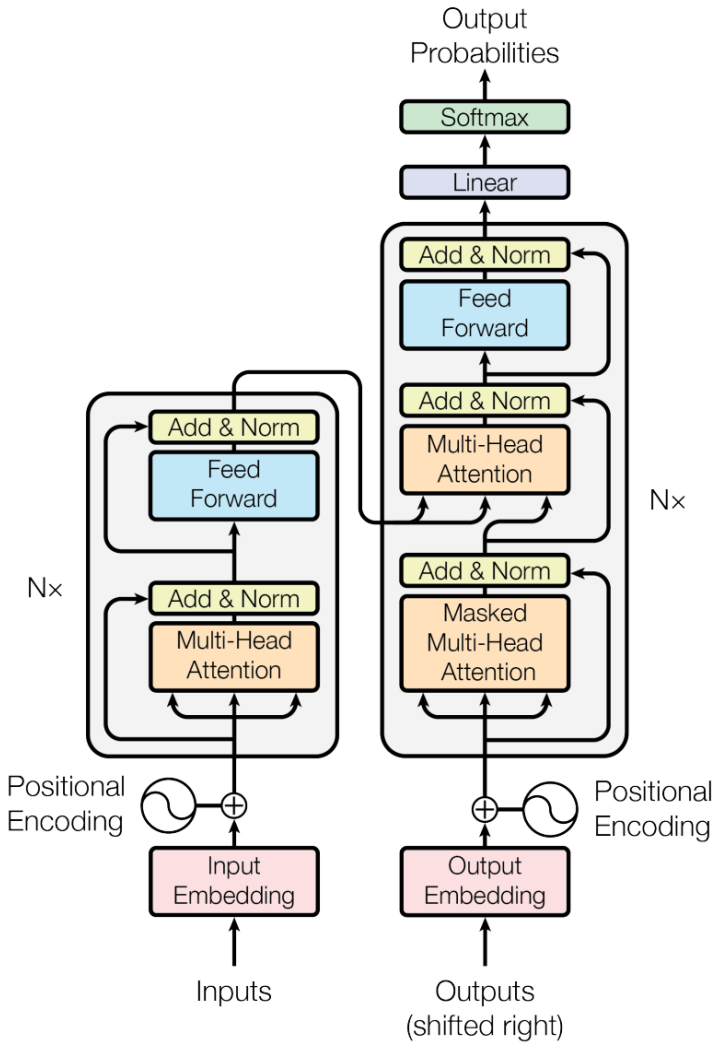
$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h) W^O$$

Dimensions: $W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$. Typically, $d_k = d_v = d_{model}/h$.

Structured Neural Networks

Transformer



Ashish Vaswani, et.al. Attention Is All You Need. NeurIPS 2017.

Structured Neural Networks

A Probabilistic Perspective of Architectures

- FNN: Weak inductive bias; assigns relatively uniform prior probability to a massive space of continuous functions.
- CNN: Strong inductive bias; assigns high prior probability to translation-equivariant and locally-correlated functions.
- RNN: Strong inductive bias; assigns high prior probability to ordered sequential dependence.
- Transformer: Weak inductive bias; inductive bias toward routing and dynamically correlating inputs regardless of spatial/temporal distance.

Structured Neural Networks

A Probabilistic Perspective of Architectures

Bayesian Marginalization: The choice of architecture dictates the "shape" of this functional space, allowing the network to naturally favor solutions that generalize well for specific data types (images, sequences, etc.)

$$p(y|x, \mathcal{D}) = \int p(y|x, W)p(W|\mathcal{D})\, dW$$

where

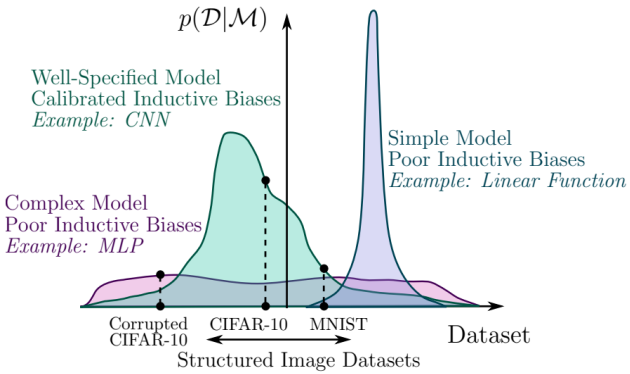
$$p(W|\mathcal{D}) \propto p(\mathcal{D}|W)p(W)$$

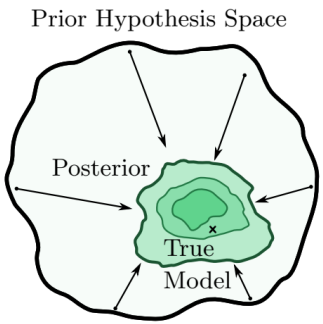
x: The input data matrix/vector.
y: The prediction (e.g., class label or continuous value).
W: The massive multi-dimensional parameter space of the architecture.
$\mathcal{D}$: The training dataset $\{(x_i, y_i)\}_{i=1}^{N}$.

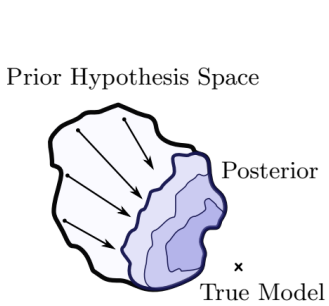
Structured Neural Networks

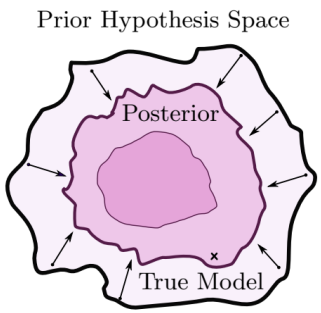
A Probabilistic Perspective of Architectures

The choice of architecture dictates the "shape" of this functional space, allowing the network to naturally favor solutions that generalize well for specific data types (images, sequences, etc.)









Andrew Gordon Wilson & Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. NeurIPS 2020.