Hands-on tinyML

A primer on Neural Networks

Francesco Paissan

University of Trento, Mila - Québec AI Institute

April 16, 2025

francescopaissan.it/tinyml-tutorial

Table of Contents

Overview

Neural Networks

Modelling

Inference

Learning

Modelling

Inference

Learning

Why care about Neural Networks?

Unless you have been living under a rock...

...you have used neural networks in your daily life.

[1] https://arxiv.org/abs/2108.11482

Why care about Neural Networks?

Unless you have been living under a rock...

...you have used neural networks in your daily life.

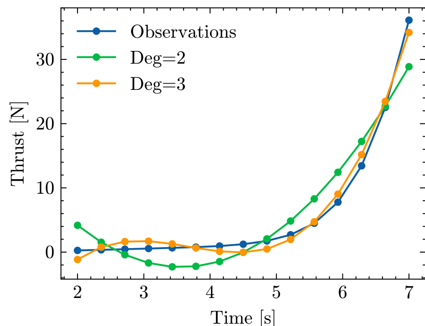
Neural networks have applications in all areas of technology. Few examples:

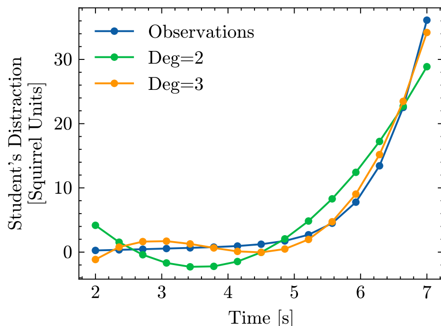
- Social Networks: Recommender systems (i.e., 'the algorithm')
- Communication: Neural Codecs
- Transportation: Graph Neural Networks (GMaps for ETA estimation)[1]
- Cybersecurity: Anomaly Detection on networks
- Manifacturing: Predictive Mantainance, Anomaly Detection

[1] https://arxiv.org/abs/2108.11482

How is it possible?

Neural networks are very versatile tools. Think of polynomial fits:





What changes between these two plots?

*Squirrel Units as in "how often a student gets distracted by a squirrel (real or metaphorical)".

Neural Networks as feature extractors

Neural Networks approximate functions[2] between vector spaces. Regardless of the:

- Units
- Input and output domains
- Underlying relationship between input and output (i.e., the groundtruth)

we can design a neural network to extract useful information from the data.

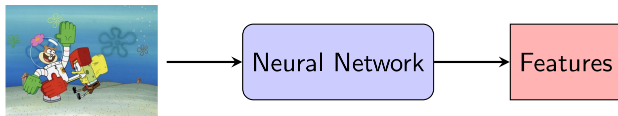
[2] Universal Approximation Theorem

Neural Networks as feature extractors

Neural Networks approximate functions[2] between vector spaces. Regardless of the:

- Units
- Input and output domains
- Underlying relationship between input and output (i.e., the groundtruth)

we can design a neural network to extract useful information from the data.



[2] Universal Approximation Theorem

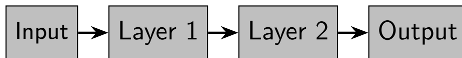
Table of Contents

Overview

Neural Networks

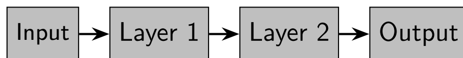
What's inside a neural network?

An ordered set of functions representing *layers*, evaluated in a cascaded fashion.



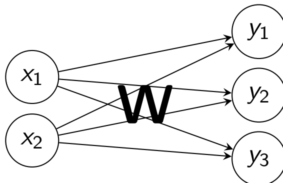
What's inside a neural network?

An ordered set of functions representing *layers*, evaluated in a cascaded fashion.



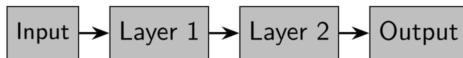
Let's define a neuron $\Lambda : \mathbb{R}^{\mathcal{D}} \rightarrow \mathbb{R}^{\mathcal{I}}$. Depending on the dimensionality of the domain and codomain ($\mathcal{D} = \{d_1, d_2, \ldots, d_D\}, \mathcal{I} = \{i_1, i_2, \ldots, i_I\}$), we can define many types of neurons:

- $\mathcal{D} = d \in \mathbb{N}, \mathcal{I} = i \in \mathbb{N}$.
  Dense layer: $\mathbf{y} \leftarrow \Lambda(\mathbf{x}) \triangleq \mathbf{x}\mathbf{W}^\top, \mathbf{W} \in \mathbb{R}^{i \times d}$.



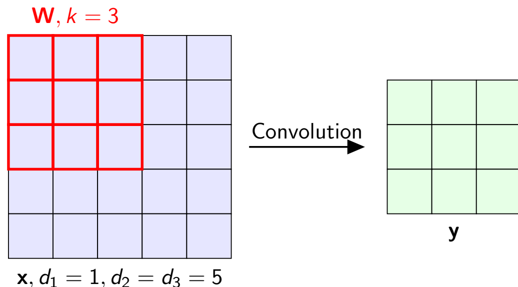
What's inside a neural network?

An ordered set of functions representing *layers*, evaluated in a cascaded fashion.



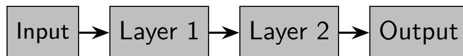
Let's define a neuron $\Lambda : \mathbb{R}^{\mathcal{D}} \rightarrow \mathbb{R}^{\mathcal{I}}$. Depending on the dimensionality of the domain and codomain ($\mathcal{D} = \{d_1, d_2, \ldots, d_D\}, \mathcal{I} = \{i_1, i_2, \ldots, i_I\}$), we can define many types of neurons:

- $\mathcal{D} = \{d_1, d_2, d_3\}, d_j \in \mathbb{N}; \mathcal{I} = \{i_1, i_2, i_3\}, i_j \in \mathbb{N}$.
  Convolutional layer: $\mathbf{y} \leftarrow \Lambda(\mathbf{x}) \triangleq \mathbf{x} * \mathbf{W}, \mathbf{W} \in \mathbb{R}^{i_1 \times d_1 \times k \times k}$.



What's inside a neural network?

An ordered set of functions representing *layers*, evaluated in a cascaded fashion.



Let's define a neuron $\Lambda : \mathbb{R}^{\mathcal{D}} \rightarrow \mathbb{R}^{\mathcal{I}}$. Depending on the dimensionality of the domain and codomain ($\mathcal{D} = \{d_1, d_2, \ldots, d_D\}, \mathcal{I} = \{i_1, i_2, \ldots, i_I\}$), we can define many types of neurons:

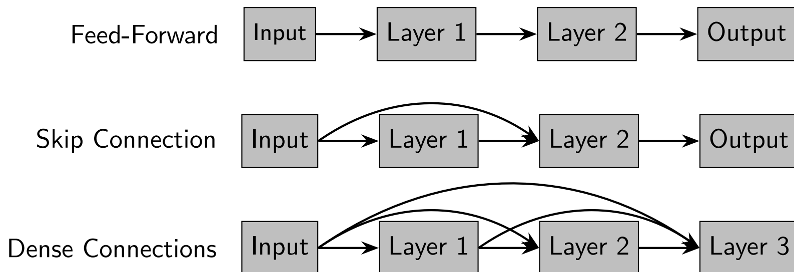
- $\mathcal{D} = \{d_1, d_2\}, d_j \in \mathbb{N}; \mathcal{I} = \{i_1, i_2\}, i_j \in \mathbb{N}, \mathbf{W}_q \in \mathbb{R}^{e \times d_1}, \mathbf{W}_k \in \mathbb{R}^{e \times d_1}, \mathbf{W}_v \in \mathbb{R}^{e \times d_1}$. Self-Attention layer:

$$\mathbf{Q} \leftarrow \mathbf{x}\mathbf{W}_q^\top, \mathbf{K} \leftarrow \mathbf{x}\mathbf{W}_k^\top, \mathbf{V} \leftarrow \mathbf{x}\mathbf{W}_v^\top,$$
$$\mathbf{y} \leftarrow \Psi(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}$$

We can compose basic layers to create more complex functions. How?

Topological structure

Neural networks exploit difference topological patterns. A few examples:



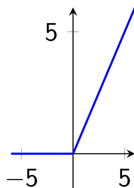
What happens when we use a feed-forward network with linear layers only?

$$\mathbf{y} = \Lambda_2 \circ \Lambda_1(\mathbf{x}) \triangleq (\mathbf{x}\mathbf{W}_1^\top)\mathbf{W}_2^\top = \mathbf{x}(\mathbf{W}_1^\top \mathbf{W}_2^\top) \implies \mathbf{y} = \Lambda_3(\mathbf{x}), \mathbf{W}_3 = \mathbf{W}_2\mathbf{W}_1. \quad (1)$$

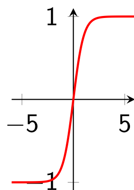
...we get another linear layer. To avoid this, we introduce non-linearities inside the network.

Non-linearities

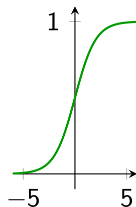
More or less any non-linear function you can think of can be used for this purpose.[3]



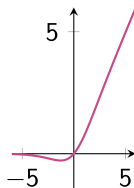
$f(x) = \max(0, x)$



$f(x) = \tanh(x)$



$\sigma(x) = \frac{1}{1+e^{-x}}$



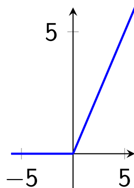
$f(x) = x \cdot \sigma(x)$

[3] The gradient of this function should be well-behaved, to avoid issues during optimization.

Non-linearities

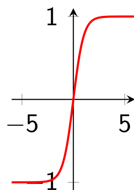
More or less any non-linear function you can think of can be used for this purpose.[3]

ReLU



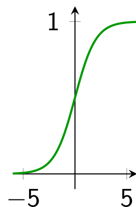
$f(x) = \max(0, x)$

Tanh



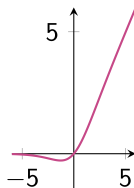
$f(x) = \tanh(x)$

Sigmoid



$\sigma(x) = \frac{1}{1+e^{-x}}$

SiLU



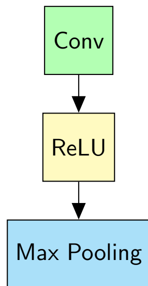
$f(x) = x \cdot \sigma(x)$

Let's review some of the most impactful neural networks through these lenses.

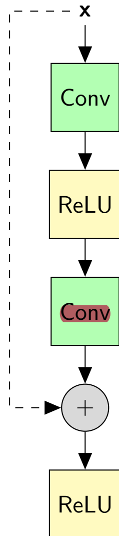
[3] The gradient of this function should be well-behaved, to avoid issues during optimization.

A couple of famous convolutional blocks

AlexNet Convolutional Block



ResNet Convolutional Block



Modelling

Inference

Learning

How do we learn?

'Learning' refers to the process of optimizing the parameters of each layer inside a neural network. ...but wait:

What should the network learn?

Learning is task-dependent

Depending on what we want to achieve, we should find a way to verify how well the neural network solves a specific task, i.e. *the error/cost/loss function*.

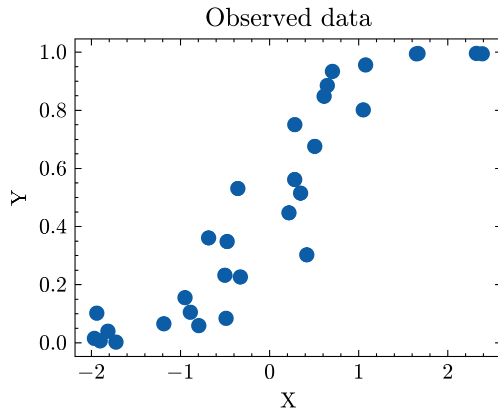
Let's create a simple use case. Let's define a single-layer neural network as:

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}^\top), \mathbf{x}, \hat{\mathbf{y}} \in \mathbb{R}^2, \mathbf{W} \in \mathbb{R}^{2\times 2}, \sigma(\mathbf{x}) \triangleq \frac{1}{1 + \exp\{-x\}} \quad (2)$$

Looks familiar?

Obtaining the data

We observe some data, and want to model it with our neural network.



We need to define the error. A reasonable choice in the case is mean squared error.[4]

[4] Why is MAE a bad idea?

Error definition

The error is a function of: the input, the output, and the weights.

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}) \triangleq \frac{1}{2}\|\sigma(\mathbf{x}\mathbf{W}^\top) - \mathbf{y}\|^2 = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|^2 \tag{3}$$

Learning is the process of adapting $\mathbf{W}$ to minimize the error. Any ideas on how we can do that?

Optimization I: Notes

The optimal weight, $\mathbf{W}^*$, represents the matrix such that

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{W}^*) = 0 \tag{4}$$

I.e., there's no error in the entire dataset. We can find (or try to find $\mathbf{W}^*$) by taking small steps towards $\mathbf{W}^*$. Let's observe that finding $\mathbf{W}^*$ translates into minimizing the error.[5] Therefore, we also know that[6]:

$$\nabla\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{W}^*) = 0 \tag{5}$$

[5] In this case, 0 represents the absolute minimum over the function's domain.

[6] Fermat's Theorem

Optimization I: Notes

The optimal weight, $\mathbf{W}^*$, represents the matrix such that

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{W}^*) = 0 \tag{4}$$

I.e., there's no error in the entire dataset. We can find (or try to find $\mathbf{W}^*$) by taking small steps towards $\mathbf{W}^*$. Let's observe that finding $\mathbf{W}^*$ translates into minimizing the error.[5] Therefore, we also know that[6]:

$$\nabla\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{W}^*) = 0 \tag{5}$$

If we compute the gradient, and it is not zero, we can do something to improve the model!

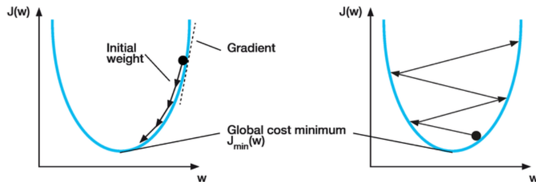
[5] In this case, 0 represents the absolute minimum over the function's domain.

[6] Fermat's Theorem

Optimization II: Gradient Descent

Intuition. Let's change the model in the most useful way:[7]

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta \nabla \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_t) \tag{6}$$



η impacts the optimization quality.

[7]Since $\nabla\mathcal{L}$ represents the direction of steepest descent.

Cross-Entropy

For classification, the model learns to predict the categorical distribution over C classes given an input $\mathbf{x}$:

$$p(c \mid \mathbf{x}) = \frac{\exp(\hat{\mathbf{y}}_c)}{\sum_{k=1}^{K} \exp(\hat{\mathbf{y}}_k)} \tag{7}$$

In this case, cross-entropy is the go-to choice[8]

$$\mathcal{L}(\mathbf{y}, \mathbf{x}) = -\sum_{c \in \mathcal{C}} \mathbf{y}_c \log p(c \mid \mathbf{x}) \tag{8}$$

[8] MLE Wiki

Fancier loss functions

You can get as fancy as you'd like, depending on the problem formulation.

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

SimCLR Loss

$$\mathcal{L}_{\text{ZS}}(\theta) = \sum_{i,j} \left\| C_{i,j} - t_i^\top f_{\text{audio}}\Big(M_\theta(t_i, h_j) \odot X_{\text{audio},j}\Big) \right\| + \lambda_1 \Big\| M_\theta(t_i, h_j) \Big\|_1 + \lambda_2 \sum_i D(X_{\text{audio},i}).$$

LMAC-ZS Loss

Modelling

Inference

Learning

What about computational complexity?

We can model the computational complexity based on non-functional constraints:

- RAM Usage: How much working memory is needed for one inference?
- FLASH Usage: how many parameters can I store?
- Latency: How many operations can I perform?[9]

[9] This is a soft constraint, we won't worry about this too much.

Let's try some of this together!

Task definition:

- CNN on CIFAR10
- Keep the computational requirements low
- Let's target a `STM32H735G-DK`, at least for simulation

Our requirements:

`Available internal RAM for AI: 560 KB | Total internal RAM: 564 KB | External RAM: 16 MB | Internal flash:  1024 KB | External flash:  64 MB.`