Predictive Calculator

Investigation of Neural Network arithmetic capabilities

Defining the project

Investigating Neural Networks for arithmetic tasks

Example of an arithmetic task:

$$1 + 2 + 3$$

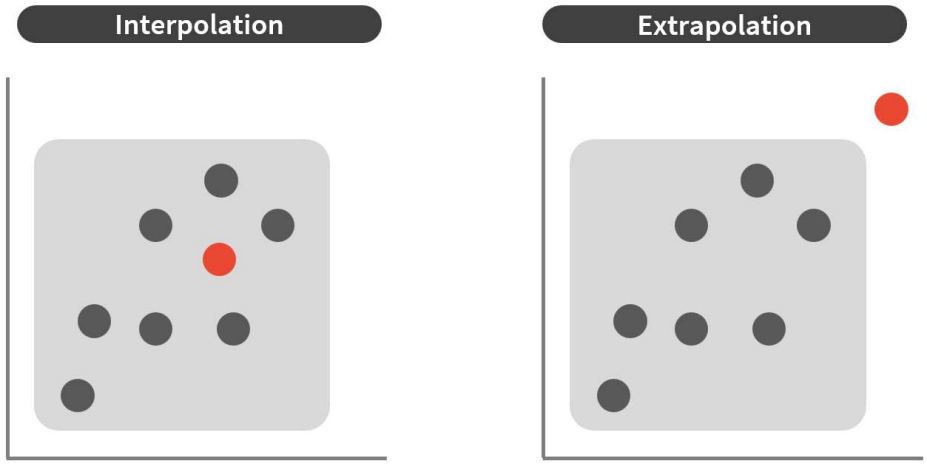


Fig. 1

Learning to solve arithmetic problems with a virtual abacus

Flavio Petruzzellis[1†], Ling Xuan Chen[1†], and Alberto Testolin[*1]

[1]University of Padova, Italy
[†]Equal contribution

Abstract

Acquiring mathematical skills is considered a key challenge for modern Artificial Intelligence systems. Inspired by the way humans discover numerical knowledge, here we introduce a deep reinforcement learning framework that allows to simulate how cognitive agents could gradually learn to solve arithmetic problems by interacting with a virtual abacus. The proposed model successfully learn to perform multi-digit additions and subtractions, achieving an error rate below 1% even when operands are much longer than those observed during training. We also compare the performance of learning agents receiving a different amount of explicit supervision, and we analyze the most common error patterns to better understand the limitations and biases resulting from our design choices.

suggest that symbolic numerical competence could emerge from domain-general learning mechanisms [7, 8]. Recent work has also tried to deploy deep reinforcement learning (RL) to solve math word problems [9]. Notably, deep RL architectures that incorporate copy and alignment mechanisms seem to discover more sophisticated problem solving procedures [10], and could even learn automatic theorem proving when combined with Monte-Carlo tree search algorithms [11].

In this work we explore whether model-free deep RL agents could learn to solve simple arithmetic tasks (expressions involving sum and subtraction) by exploiting an external representational tool, which can be functionally conceived as a *virtual abacus*. Differently from the above-mentioned approaches, our goal is to take advantage of the way humans solve the task by simulating the interaction of the RL agent with an abacus, which can be

Why is this interesting?

- Neural networks struggle with arithmetic problems
- Mathematical cognition would be useful to Neural Networks

Quick refresher: Neural networks

Variables, which the Neural Network can tweak:

- Weights
- Biases
- a – parameter in the activation function

$$x_j = f\left(\sum_i (w_{ji} x_i) + b_j\right)$$

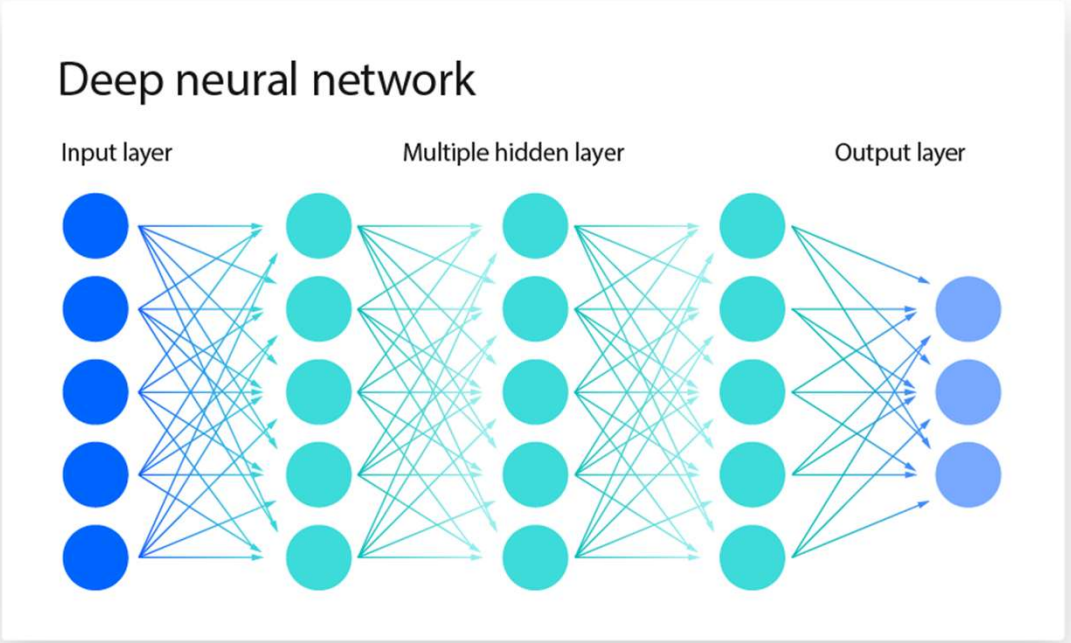


Fig. 2

Literature Review

Neural Network Arithmetic Capabilities: a Literature Review

Anton Mukin

May 2025

Contents

Literature Review

Under a fixed budget for training-data and parameters, which neural-network architectures demonstrate the best systematic generalization on elementary arithmetic tasks (n-digit addition, subtraction, multiplication, and division) when evaluated on operand lengths and magnitudes that fall outside the training distribution, without relying on hand-coded or symbolic rules or other deterministic systems?

Interesting finds and Takeaways

- Continuous Nature of Neural Networks
- The impact of nonlinear activation functions
- External memory approach
- Activation function: PReLU
- Preprocessor: seq2grid
- Recurrent Neural Networks
- Transformers
- Pre- Trained Large Language Model

First Prototype

- Feed-forward Neural Network
- Number Range: (-5, 5)
- Operators: (+, -)
- Example from training array: (3 + 4 + -4)

First Prototype

```
model = keras.Sequential([
keras.Input(shape=(len(x[0]),)),
layers.Dense(32),
PReLU(),
layers.Dense(32),
PReLU(),
layers.Dense(1,  activation = "linear")
])

model.compile(optimizer="adam", loss="mse", metrics=["mae", "mse"])

train_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(32)
val_dataset = tf.data.Dataset.from_tensor_slices((x_val, y_val)).batch(32)

history = model.fit(
train_dataset,
validation_data=val_dataset,
epochs=200,
callbacks=[early_stopping]
)
```

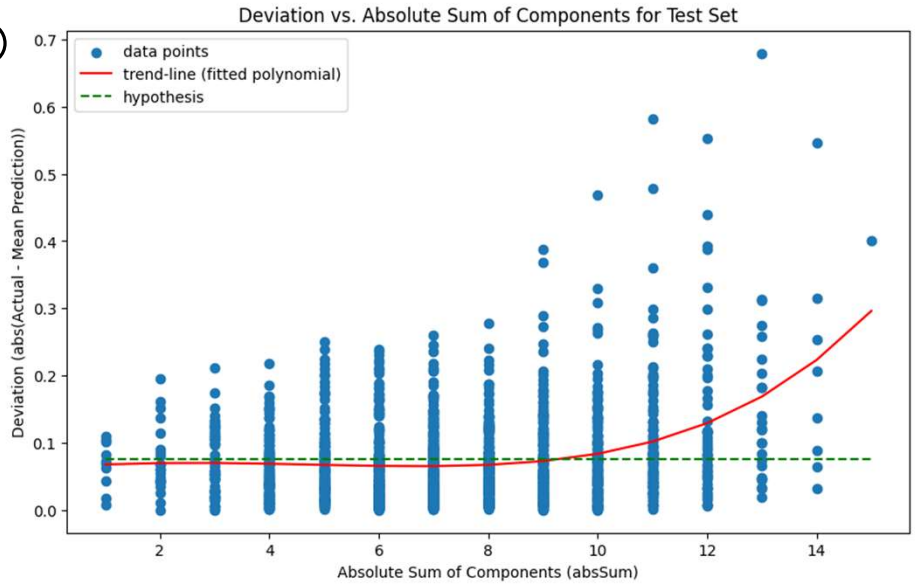
First Prototype

- Expressions not in x: (3 + 3 – 2)
- Expressions outside of the number range: (8 + 8 – -7)
- Expressions with differing length: (3 + 3 – 2 + -1)

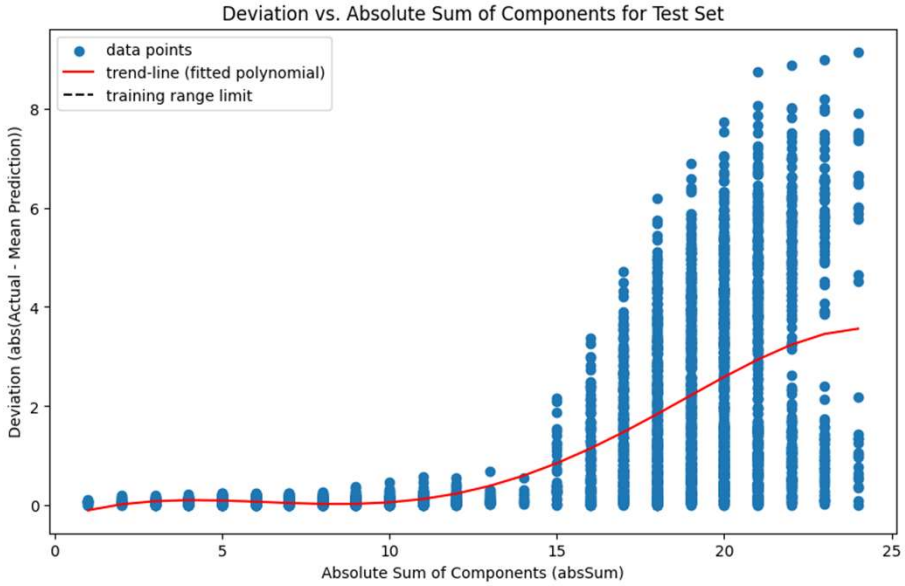
Results

① expressions inside the number range
② expressions outside the number range
③ longer expressions

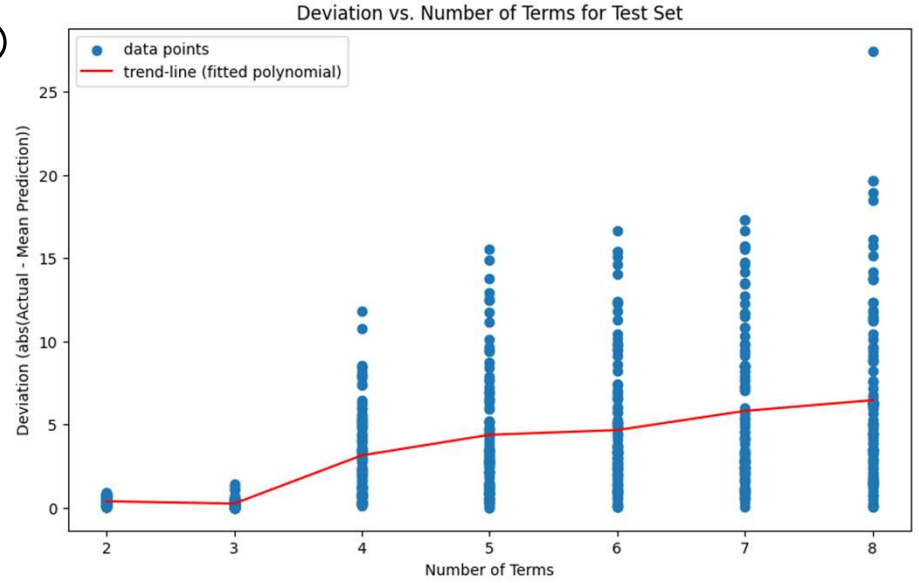
①



②



③



Benchmark

Scales with:

- MSE inside number range
- MSE outside number range
- MSE for longer expressions

the results of the basemodel FNN will be used as baseline.

MSE inside number range: 0.0058327843

MSE outside number range (absSum= 22): 10.093028

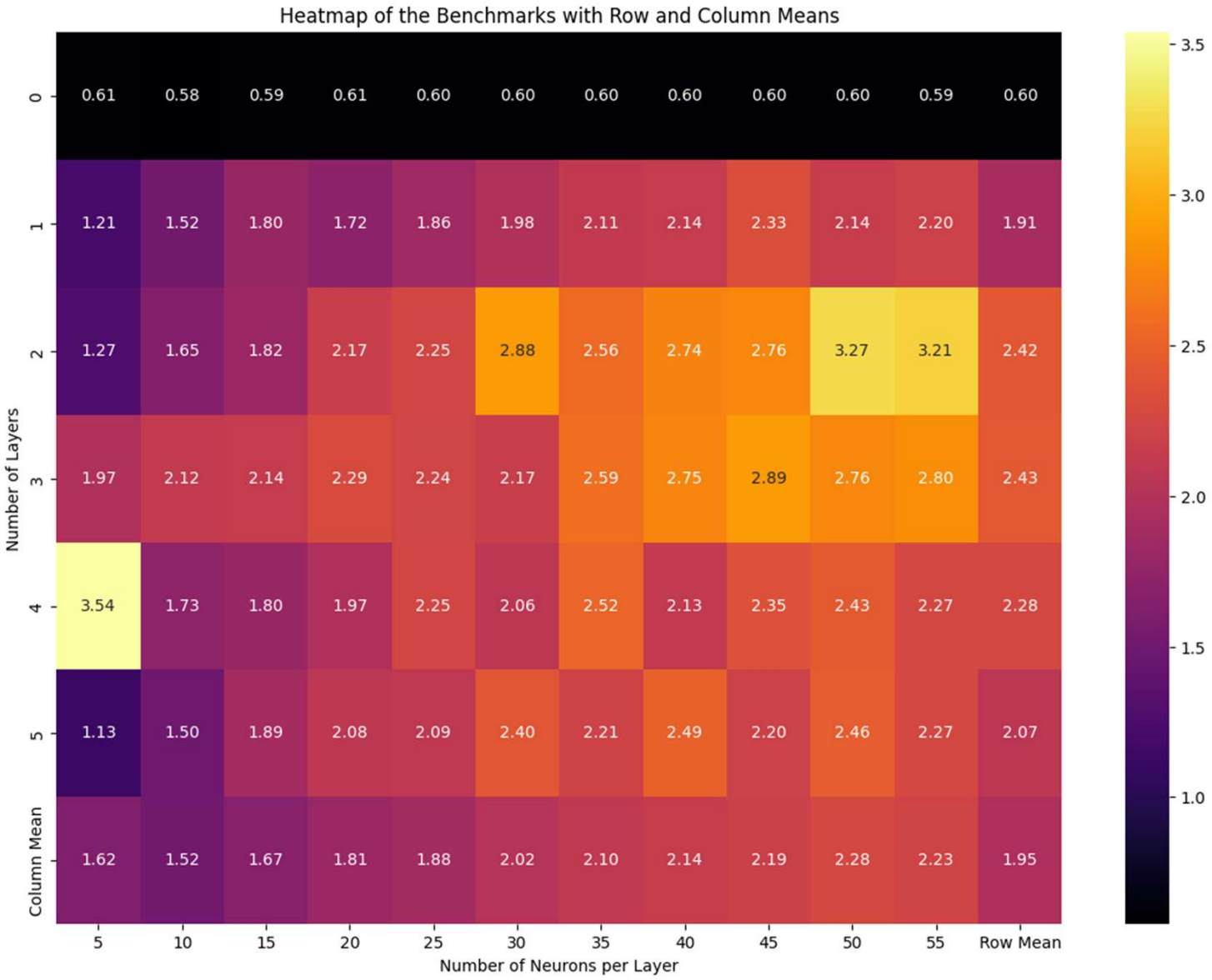
Deviation for longer expressions: Deviation for expressions with 4 numbers: 10.760098

all 3 Values will be weighed equally as important.

The benchmark value is the sum

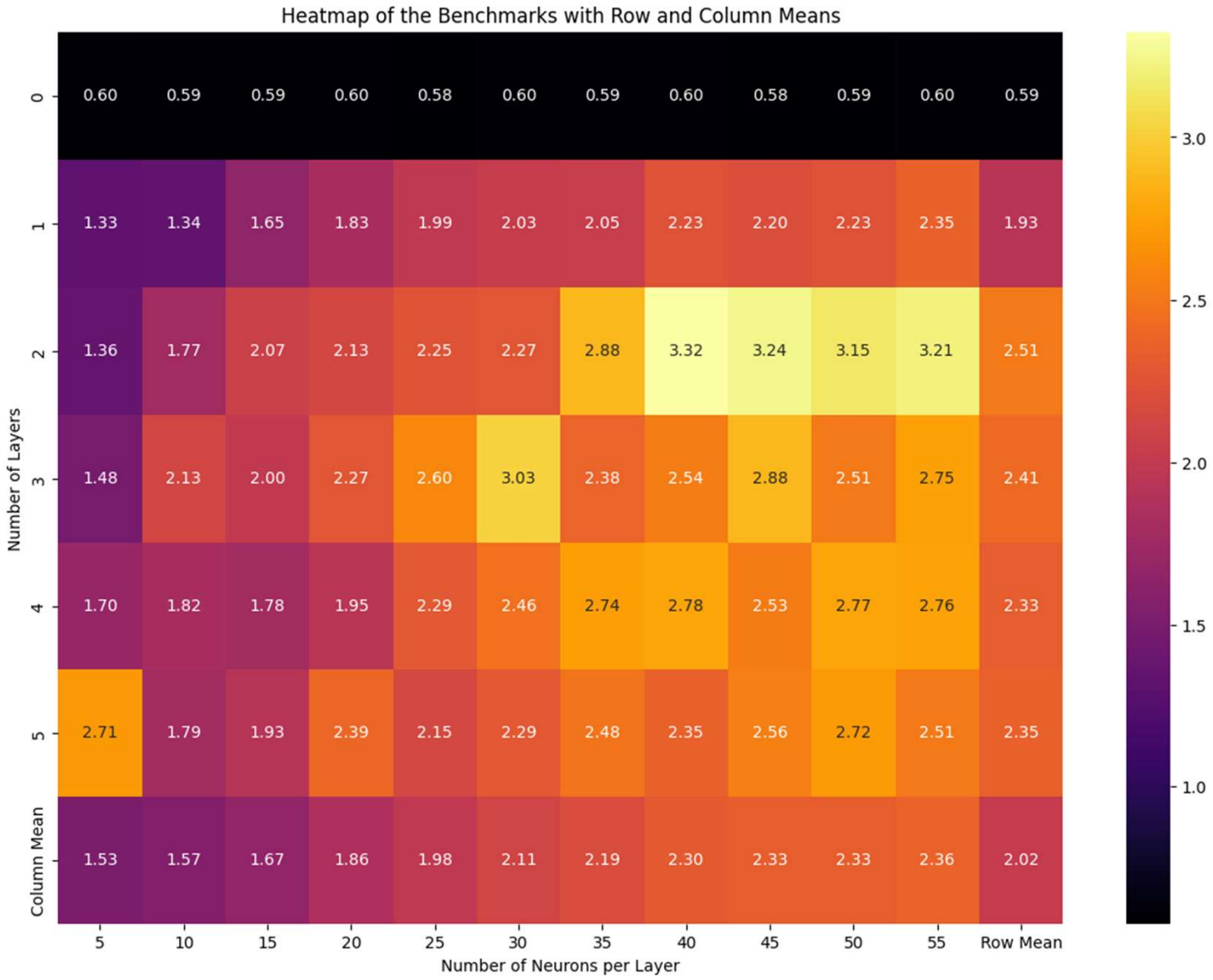
Results

Analysis of the optimal number of layers and neurons per layer



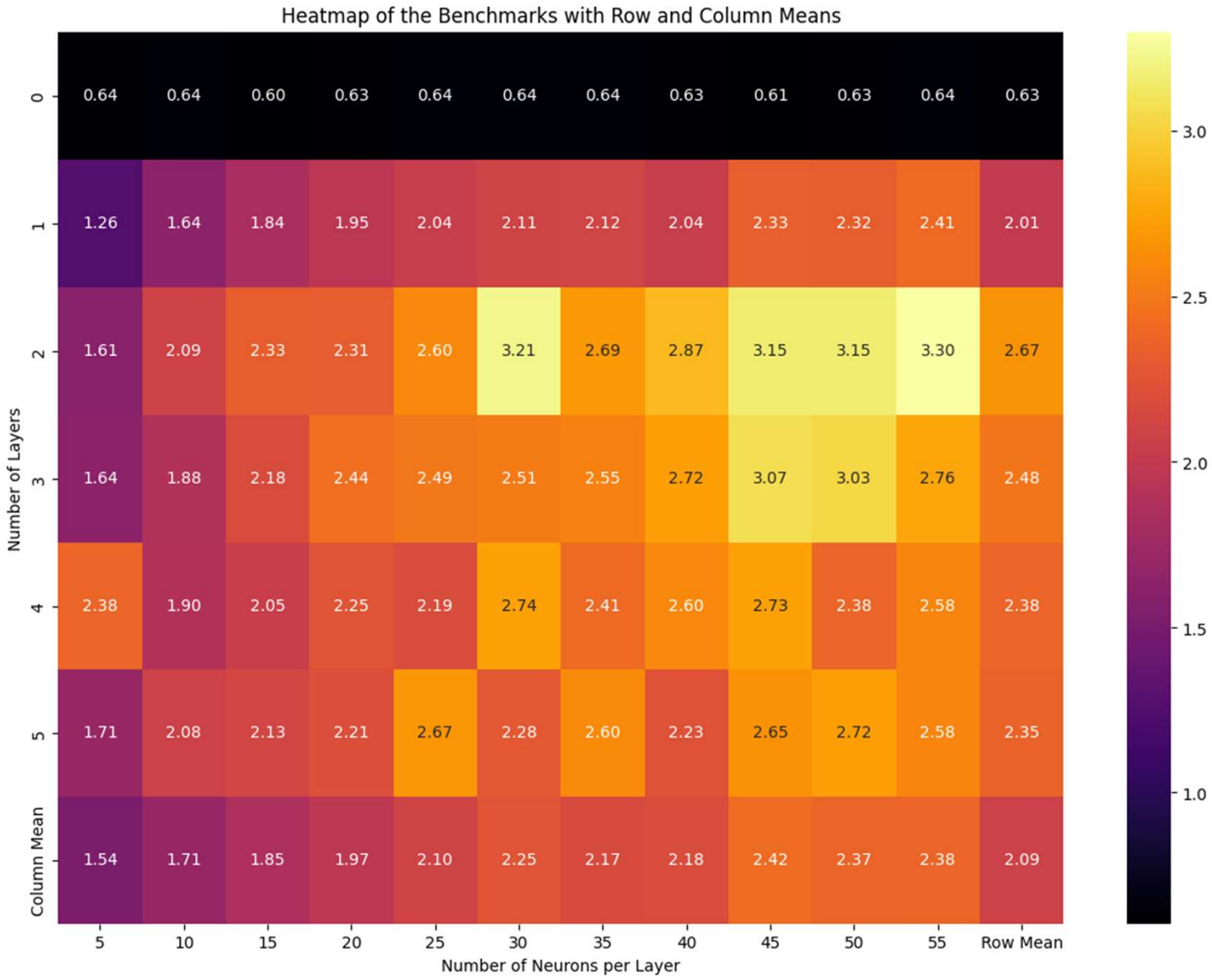
Results

Analysis of the optimal number of layers and neurons per layer



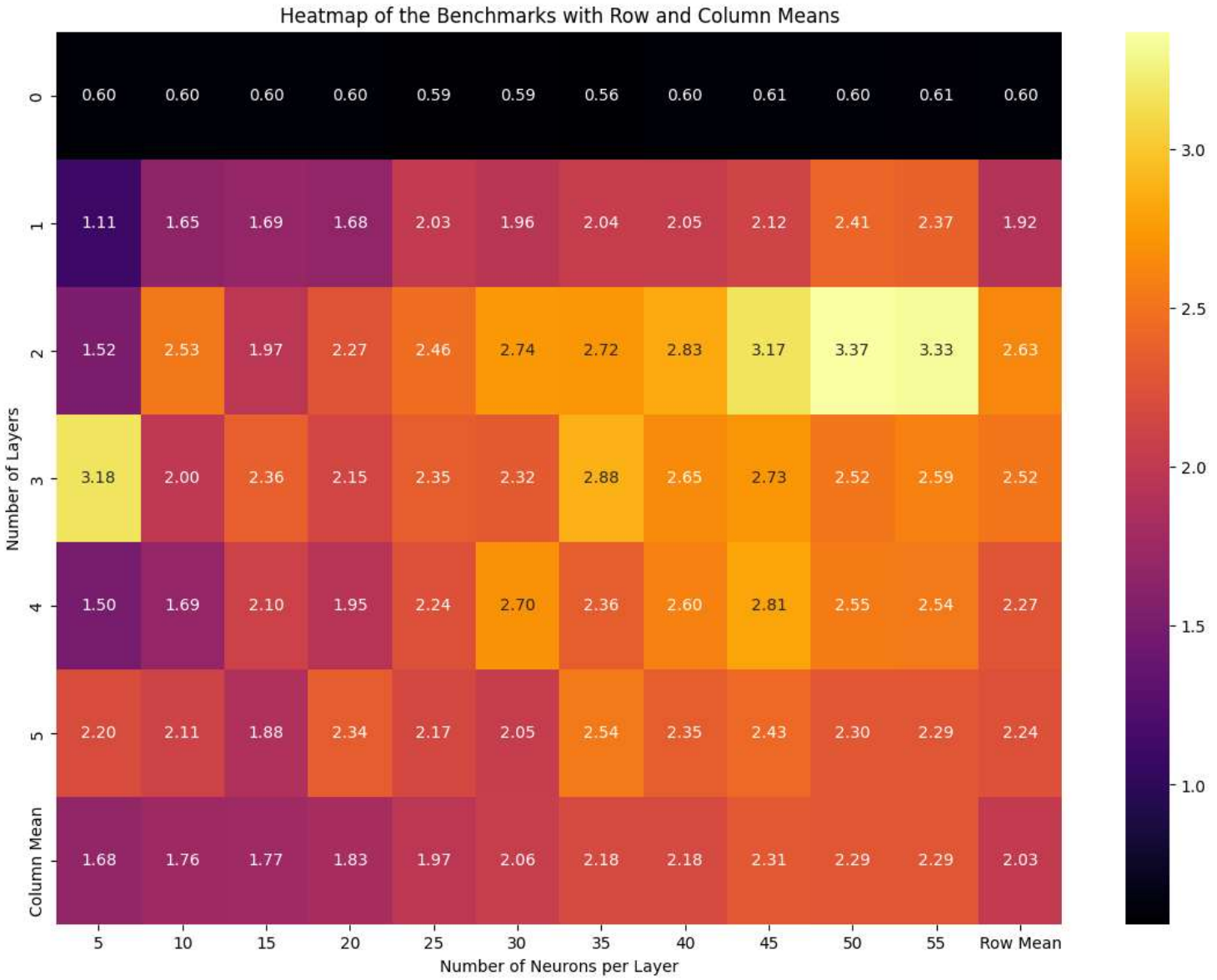
Results

Analysis of the optimal number of layers and neurons per layer



Results

Analysis of the optimal number of layers and neurons per layer



What to do next?

- Regression?
- Drop out.
- batch sizes?
- The role of each individual layer/neuron?
- Try out the Nvidia Jetson Orin Nano GPU?
- Try out other Architectures?

Closing Remark

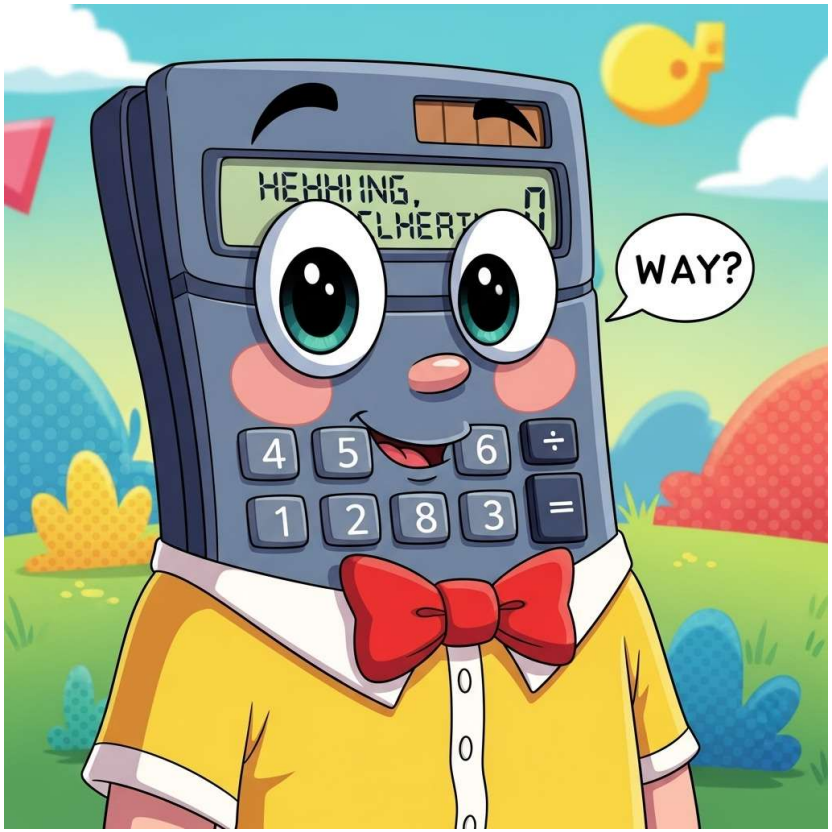


Fig. 3

Fig. 4

Back Up Slides

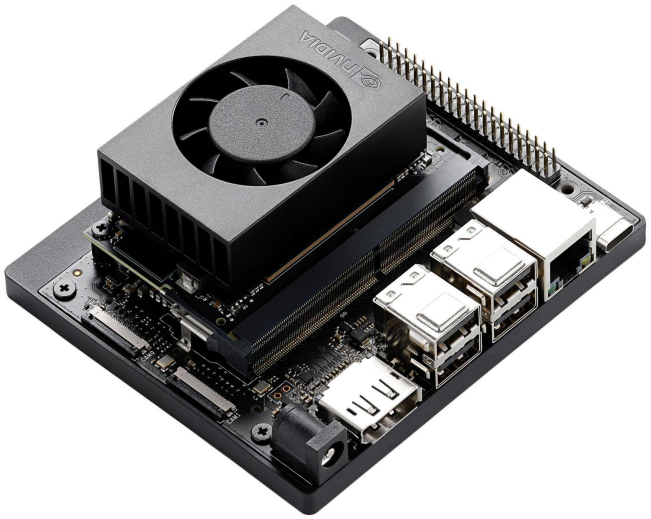
Fig. 1 source: https://towardsdatascience.com/the-machine-learning-guide-for-predictive-accuracy-interpolation-and-extrapolation-45dd270ee871/

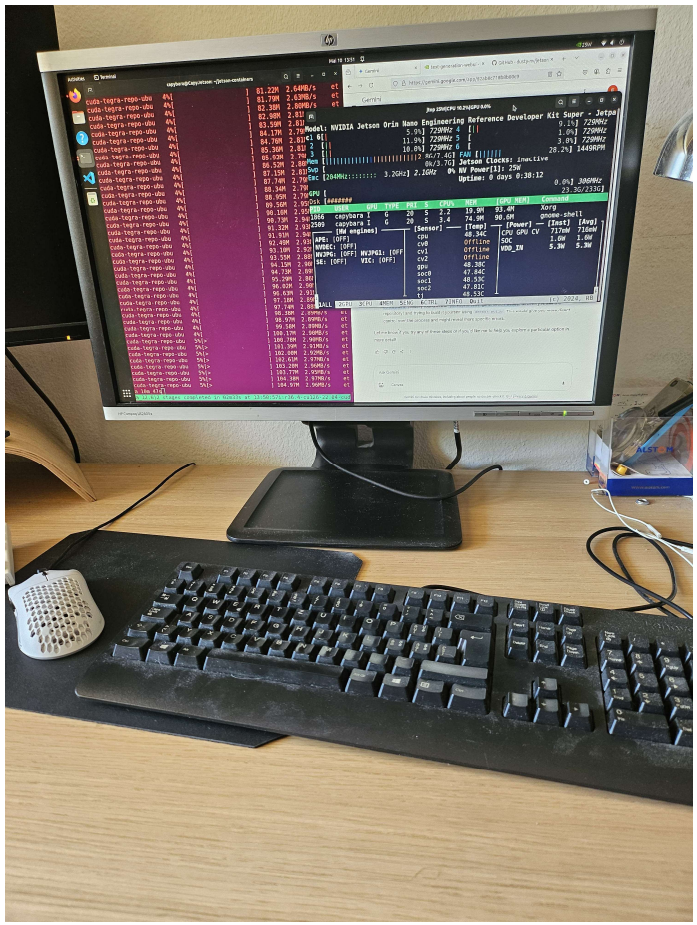
Fig. 2 source: https://www.ibm.com/think/topics/neural-networks

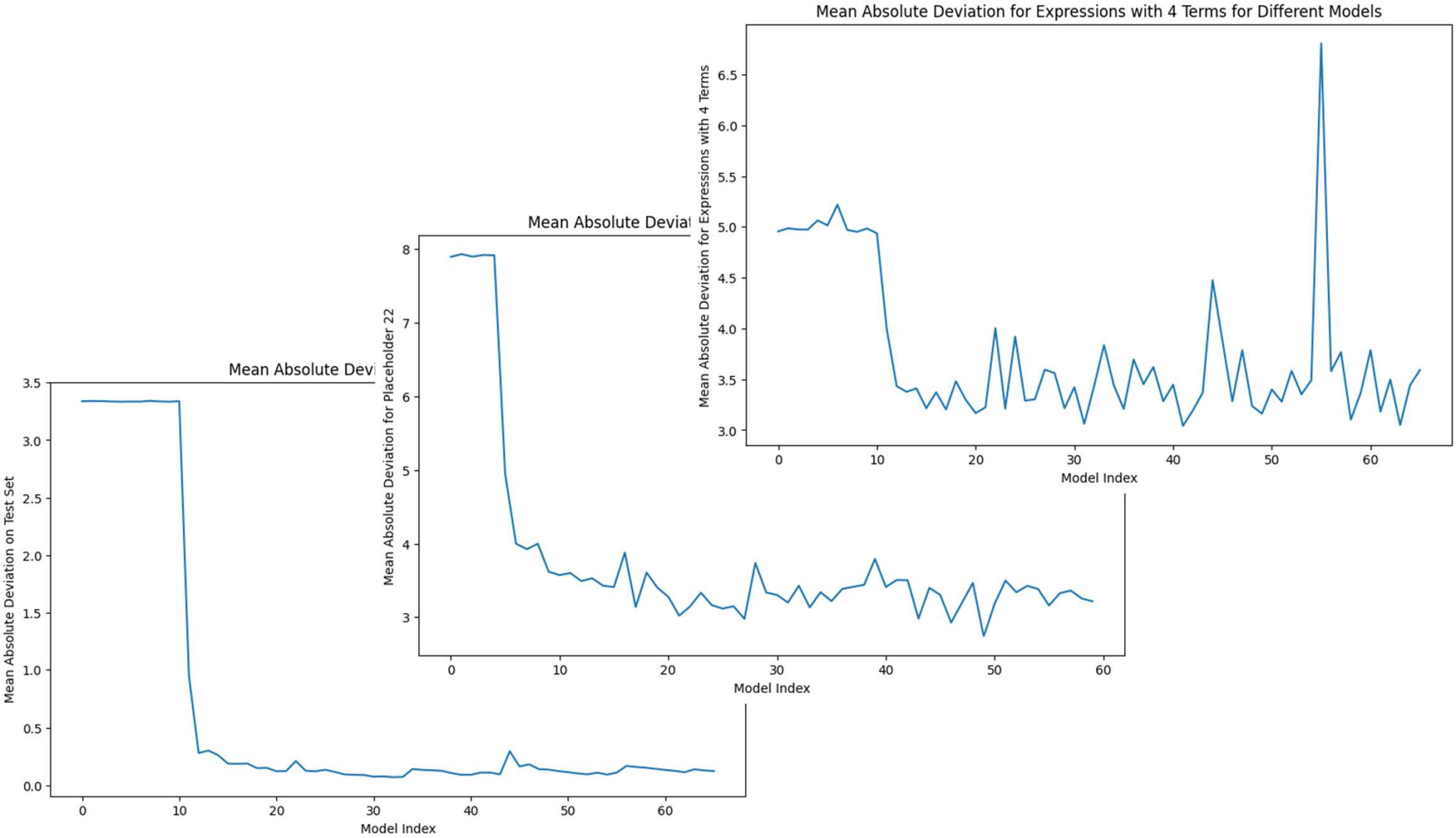
Fig. 3 source: generated with Google Gemini. Prompt: Draw a personified talking calculator character in a cartoon style

Fig. 4 source: generated with Google Gemini. Prompt: Draw a personified talking calculator character in a cartoon style

Nvidia GPU






Mean Absolute Deviation for Expressions with 4 Terms for Different Models
Mean Absolute Deviation for Expressions with 4 Terms
Model Index
Mean Absolute Deviation for Placeholder 22
Model Index
Mean Absolute Deviation on Test Set
Model Index

Tools

- Overleaf
- Google colab (Jupyter Notebooks)
- Tensorflow (keras library)

Complications

- Reproducibility issues
- Interpretation of the graphics is inconsistent

Reflection

- Better definition of research question
- Clearer code (use more comments)

Workflow

1. Research
2. Setting up work environment in Nvidia Jetson
3. Literature Review
4. First Prototype

Motivation for this project

- Systematic Calculators < Physics Informed Neural Networks (PINN)
- Kaggle.com
- Previous individual Projects

Activation Functions

- PReLU
- Sigmoid
- Tanh