Vision Transformer (ViT)

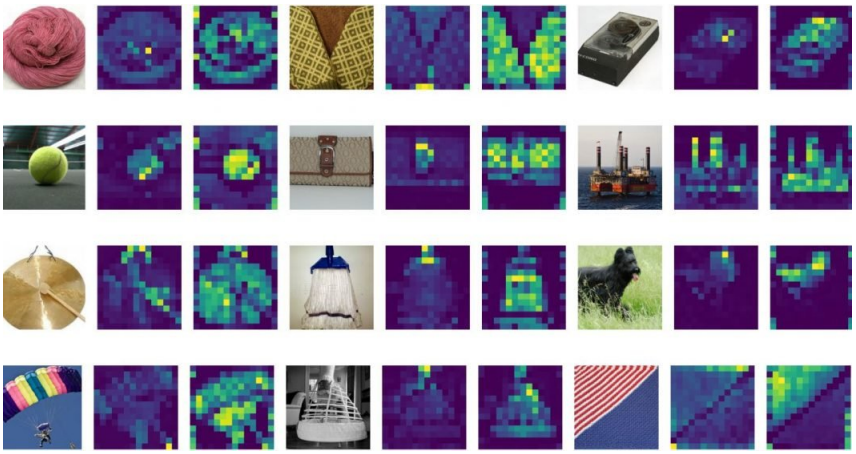
Roxana Kramer, Miruna Sapca, Marian Ostate, Victor Gherghel

West University of Timisoara, Faculty of Mathematics and Computer Science

Abstract. We choosed the Vision Transformer (ViT) model for our project, drawn to its innovative approach and promising capabilities. Our decision to choose ViT comes from a convergence of factors that align perfectly with the distinctive requirements and targets of our project. The big interest and ongoing support within the research community for ViT helped us with making our project decision. This ensures that we are working with a model backed by many collective knowledge. In summary, our choice of ViT is influnced by a colection of its unique features and adaptability to project requirements.

1 Benchmark

Vision Transformer (ViT) is a groundbreaking deep learning architecture that has revolutionized computer vision tasks, departing from traditional convolutional neural networks (CNNs). Introduced by researchers at Google in 2020, ViT leverages the power of transformers, originally designed for natural language processing, to process image data in a highly efficient and scalable manner.



The structure of the vision transformer architecture consists of the following steps:

1. Split an image into patches (fixed sizes)
2. Flatten the image patches
3. Create lower-dimensional linear embeddings from these flattened image patches
4. Include positional embeddings
5. Feed the sequence as an input to a state-of-the-art transformer encoder
6. Pre-train the ViT model with image labels, which is then fully supervised on a big dataset
7. Fine-tune the downstream dataset for image classification

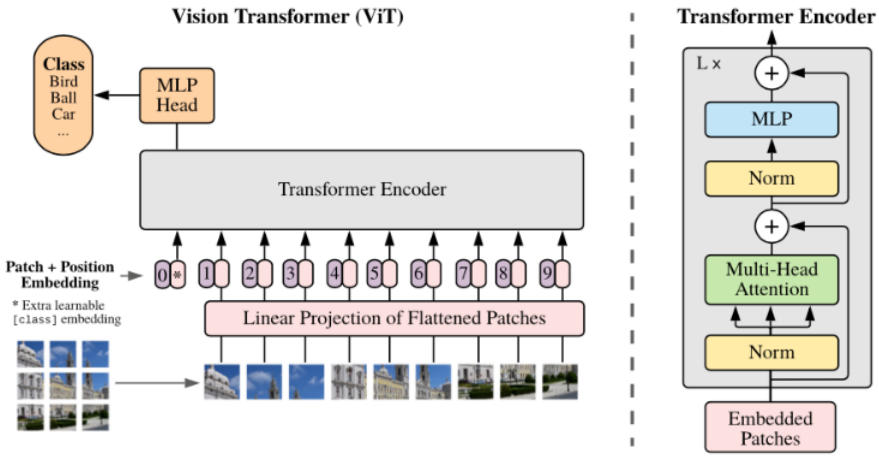
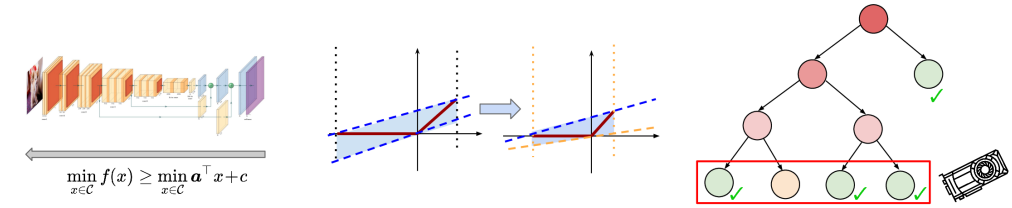


Fig. 1. Vision Transformer ViT Architecture

2 Tools

2.1 alpha-beta-CROWN

$\alpha, \beta-$CROWN functions as a neural network verifier employing a streamlined linear bound propagation framework and branch-and-bound techniques. Its computational efficiency is enhanced when deployed on GPUs, allowing effective scaling to sizable convolutional networks with millions of parameters. The alpha-beta-CROWN method is capable of offering provable assurances of robustness against adversarial attacks while also verifying other general properties inherent in neural networks.



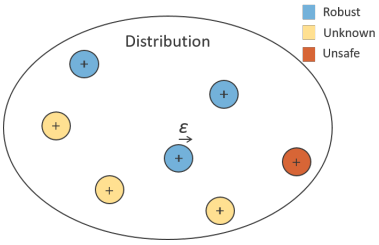
1. CROWN is a fast verification method that uses a backward propagation of linear inequalities through a neural network, relaxing activation functions with linear bounds.
2. $\alpha-$CROWN optimizes CROWN for the Fast-and-Complete verifier, improving intermediate and final layer bounds through a variable α. It's more powerful than LP due to its ability to refine intermediate layer bounds inexpensively.
3. $\beta-$CROWN introduces an optimizable parameter β to include split constraints in branch and bound into the CROWN process. This creates a robust and scalable neural network verifier by combining branch and bound with efficient GPU-accelerated bound propagation.

2.2 PyRat

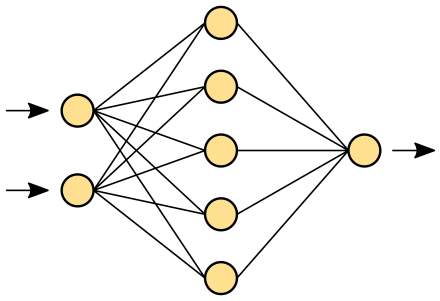
PyRAT (Python Reachability Assessment Tool) is an open source tool used to examine a deep neural network's state reachability. Its foundation is abstract interpretation, a method for approximating programme execution routes in order to reason the programme behaviour. PyRAT seeks to determine if a neural network can achieve any of the goal states from any given input by using a collection of target states and the neural network as input.

PyRAT's uses:

1. tool for safety testing, since it can confirm whether a neural network is not exhibiting any unwanted behaviours, such crashing or giving inaccurate outputs.
2. analysis of robustness: PyRAT may be used to examine how resilient a neural network is against hostile inputs.



3. explainability: The decisions made by a neural network may be explained using PyRAT.



Practical examples of uses include confirming that a neural network used for medical diagnosis does not generate false diagnoses, examining how resilient a neural network is to hostile inputs, and explaining the reasoning behind a neural network's image decision-making.

Key characteristics of PyRAT make it stand out as a powerful tool for neural network research. It's wide support for a variety of neural network topologies, including feedforward, convolutional, recurrent, and deep reinforcement learning networks, is its first noteworthy characteristic. PyRAT's adaptability to the difficulties included in different model designs is ensured by its flexibility. PyRAT's exceptional accuracy in determining whether a neural network can achieve a given state is another noteworthy feature that sets it apart from the competition and gives users trustworthy information about the capabilities and constraints of their models. Moreover, PyRAT stands out for its efficiency, especially with big neural networks. This efficacy not only simplifies the analysis procedure but also establishes PyRAT as a useful tool for practitioners and academics dealing with intricate and large-scale models. To summarise, PyRAT is an effective tool for neural network analysis because of its accuracy, efficiency, and support for a variety of designs.

3 Installation and Usage of $\alpha, \beta-$CROWN

```
# Create directory and clone repositories

mkdir abc_tmp
cd abc_tmp
git clone https://github.com/Verified-Intelligence/alpha-beta-CROWN.git
git clone https://github.com/Verified-Intelligence/auto_LiRPA.git in abcrown/
    autolirpa
git clone https://github.com/ChristopherBrix/vnncomp2023_benchmarks.git in
    abc_tmp

cd vnncomp2023_benchmarks ; ./setup.sh -

# which will unzip all the vnnlib and onnx archives

# Clone and install miniconda (conda will also work)

cd abc_tmp
```

```
install miniconda + all the features
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh

# restart console - otherwise it will not work

# Create the alpha-beta-crown environment

cd abcrown/complete verifier
# Remove the old environment, if necessary.
conda deactivate; conda env remove --name alpha-beta-crown

# Install all dependencies into the alpha-beta-crown environment

conda env create -f complete_verifier/environment.yaml --name alpha-beta-crown

# Activate the environment

conda activate alpha-beta-crown
cd complete_verifier

# Try cifar_resnet_2b.yaml to see if alpha-beta-crown is working properly

# Because it is a VM with no dedicated GPU, in order to bypass the "Not enough memory"
# or "No CUDA GPUs available" we must use the \textbf{--device cpu} flag
python abcrown.py --config ./exp_configs/vnncomp23/cifar_resnet_2b.yaml --device cpu

# Result: safe incomplete in 3.5271 seconds, check Cifar_resnet_2b results.txt in

https://github.com/VictorGhe/VF-proiect/blob/main/alpha-beta-crown/

# Try vit.yaml to see if alpha-beta-crown is working properly
python abcrown.py --config ./exp_configs/vnncomp23/vit.yaml --device cpu

# Results: check "VITresults", "VITresults#2", "VITresults#3" in
https://github.com/VictorGhe/VF-proiect/tree/main/alpha-beta-crown
```

References

1. Benchmark: https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main/benchmarks/vit
2. Tool 1: https://github.com/Verified-Intelligence/alpha-beta-CROWN
3. Tool 2: https://github.com/ChristopherBrix/vnncomp2023_results/tree/main/pyrat
4. https://viso.ai/deep-learning/vision-transformer-vit/?fbclid=IwAR1lTOAfng_T7diBYAgxWzjSKSpHhNLDBpXwawuCIfRWoP5IMZD1Ufd3GCc
5. https://pyrat-analyzer.com
6. https://github.com/pyratlib/pyrat