Aalto-yliopisto
Perustieteiden
korkeakoulu

Lossless Compression of Deep Neural Networks (results-presentation)

Vilhelm Toivonen

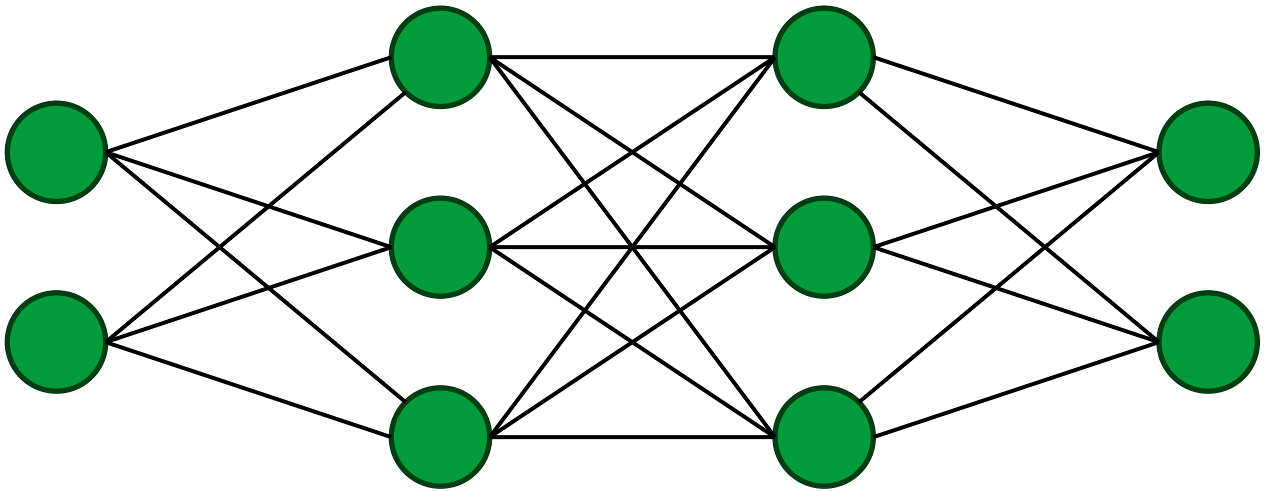
01.12.2023

Instructor: Nikita *Belyak*

Supervisor: Fabricio *Oliveira*

Systeemianalyysin
laboratorio

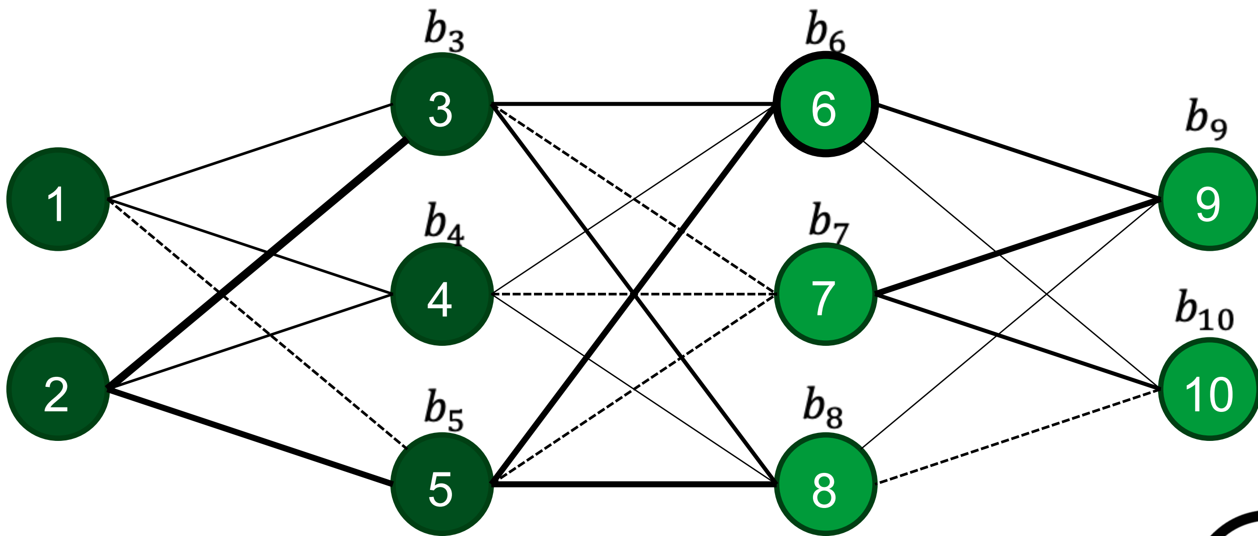
Background – DNNs



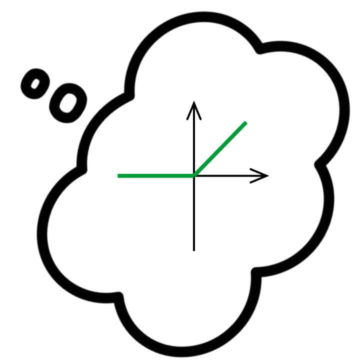
Larger DNNs make calculations more intensive

- Slow forward passes
- Computationally expensive to create mathematical programming problems

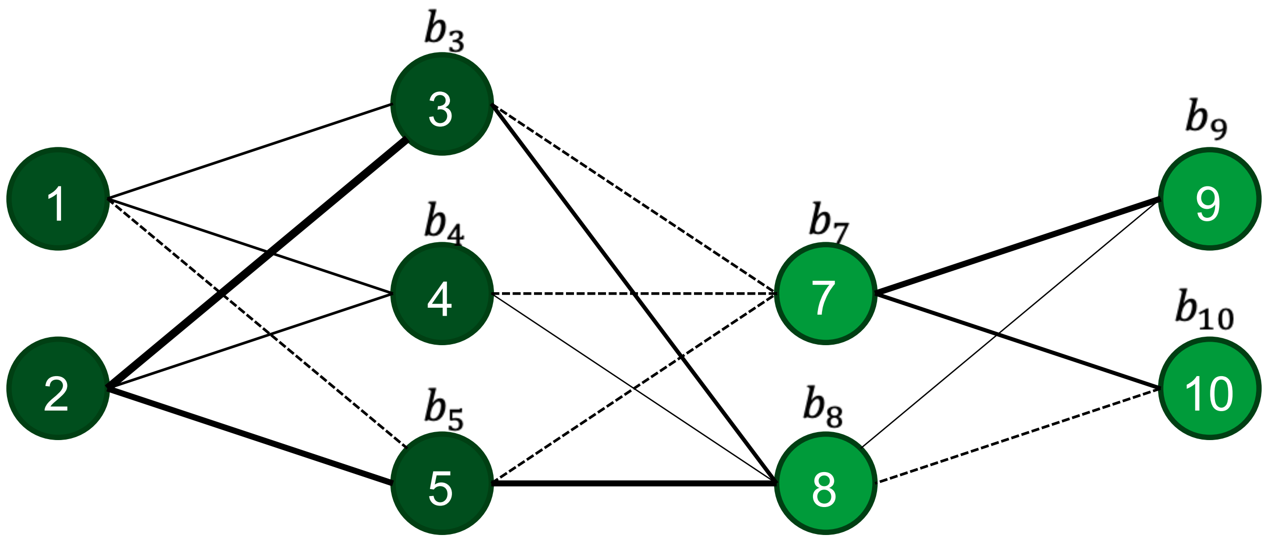
Example – Upper bound



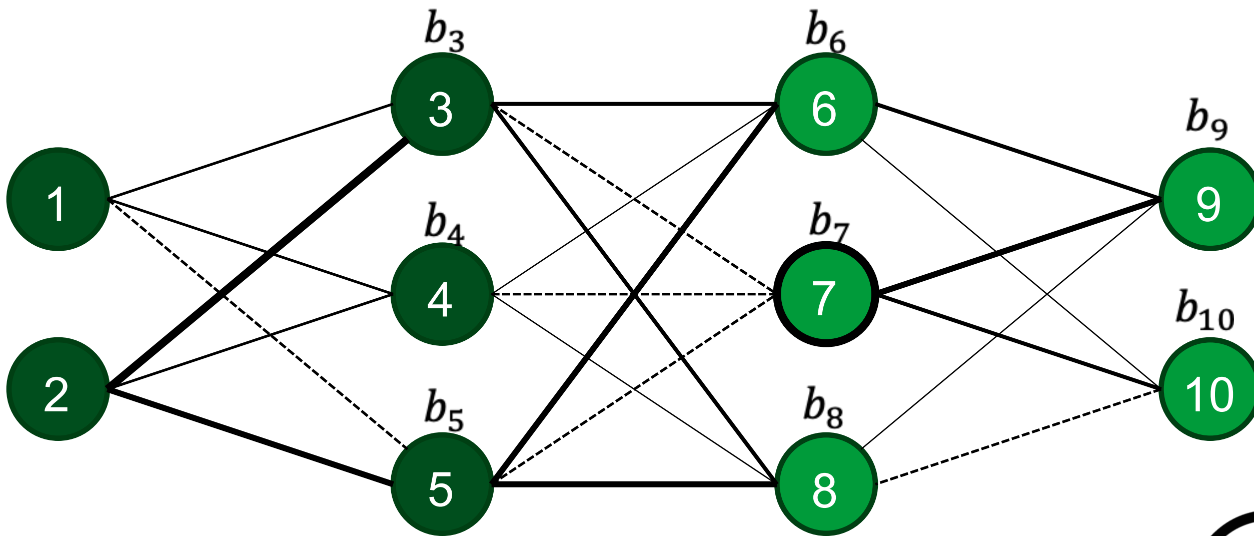
$$l = 2$$
$$i = 1$$
$$G_i^l < 0$$



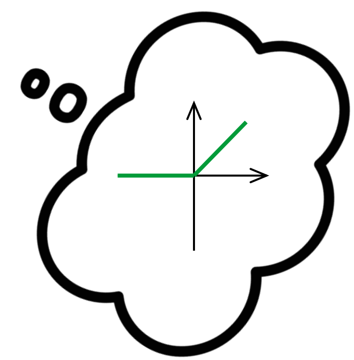
Example – Upper bound



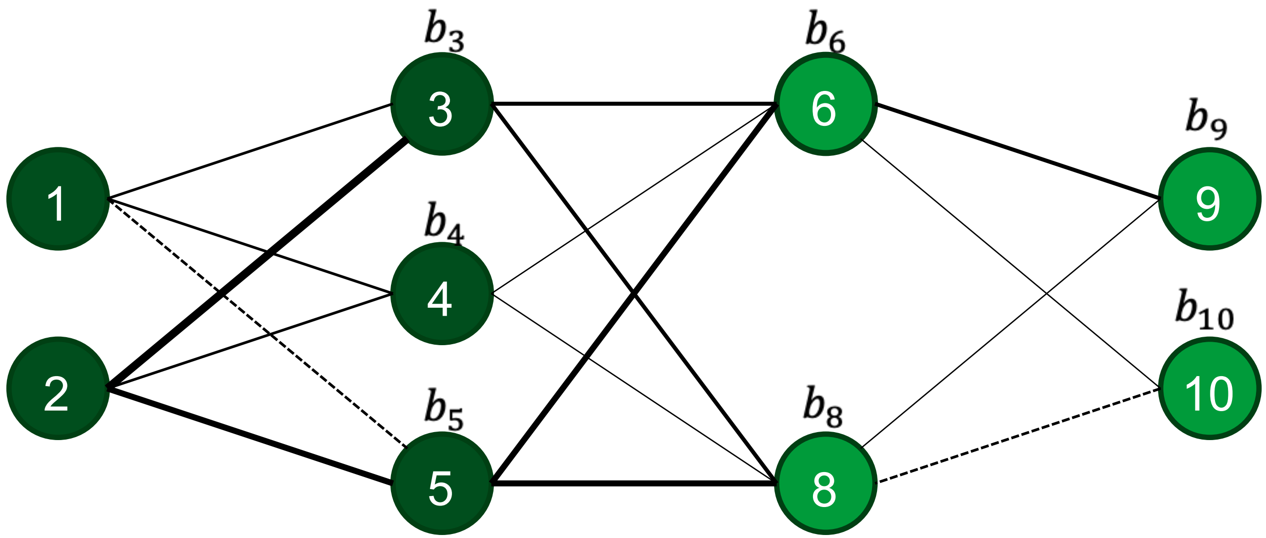
Example – Zero weights



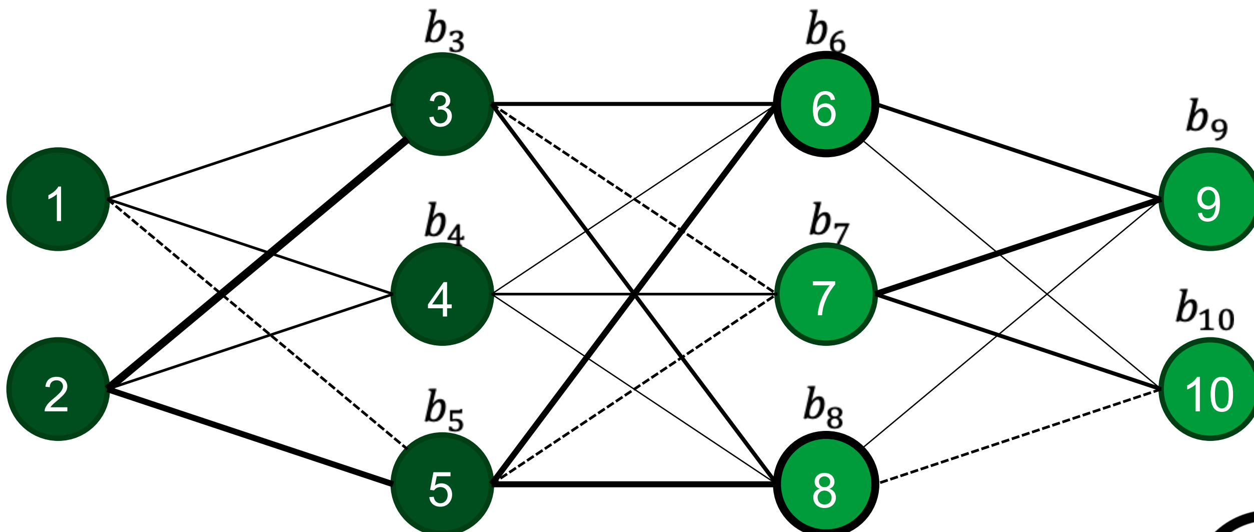
$$l = 2$$
$$i = 2$$
$$W_i^l = 0$$



Example – Zero weights

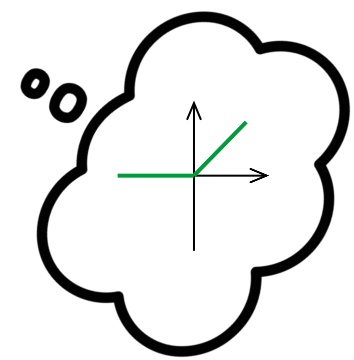


Example – Linear dependence

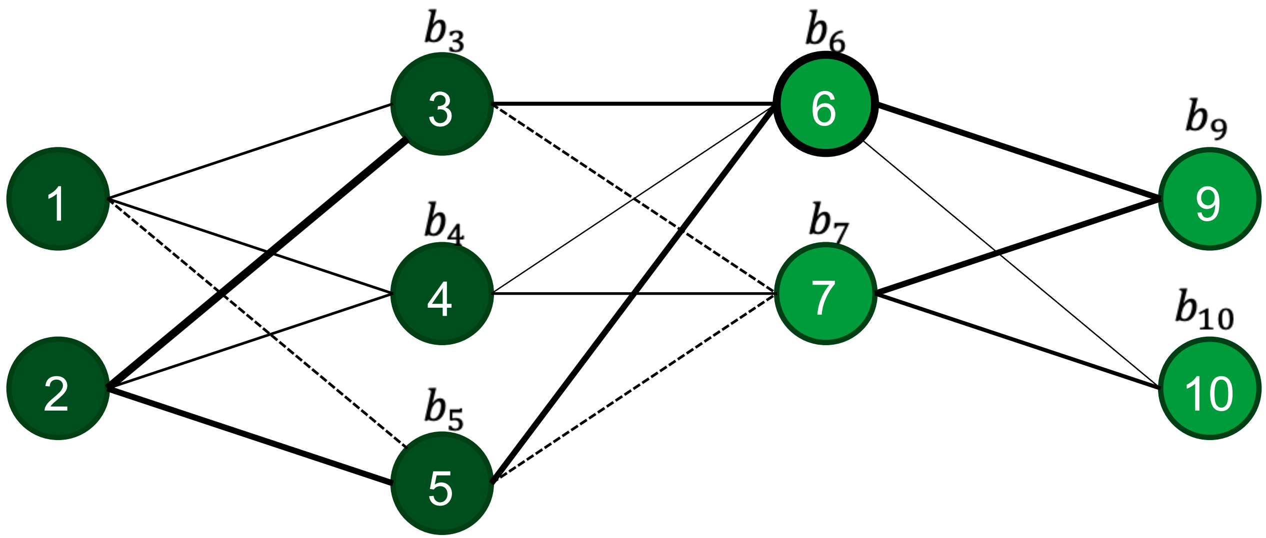


$l = 2$
$i = 1$
$\bar{G}_i^l > 0$

$l = 2$
$i = 3$
$\bar{G}_i^l > 0$



Example – Linear dependence



Aalto-yliopisto
Perustieteiden korkeakoulu

Systeemianalyysin laboratorio

Methods

- Train the models using different optimizers and regularizations
- Find the upper bounds G and the lower bounds $\bar{G}$ for each neuron
- Prune the network with the illustrated algorithm
- Check that the output of the pruned network matches the output of the original network, and log results

for $l \leftarrow 1, \ldots, L$ **do**
  $S \leftarrow \{\}$ ▷ Set of stable units left in layer l
  Unstable ← False ▷ If there are unstable units in layer l
  for $i \leftarrow 1, \ldots, n_l$ **do**
    if $G_i^l < 0$ for $x \in \mathbb{D}$ or $W_i^l = 0$ **then** ▷ Stabily inactive, constant output
      if $i < n_l$ or $|S| > 0$ or Unstable **then**
        if $W_i^l = 0$ and $b_i^l > 0$ **then**
          for $j \leftarrow 1, \ldots, n_{l+1}$ **do**
            $b_j^{l+1} \leftarrow b_j^{l+1} + w_{ji}^{l+1} b_i^l$
          end for
        end if
        Remove unit i from layer l ▷ Unit i is not necessary
      end if
    else if $\bar{G}_i^l > 0$ for $x \in \mathbb{D}$ **then** ▷ Stabily active
      if $\mathrm{rank}(W_{S \cup \{i\}}^l) > |S|$ **then**
        $S \leftarrow S \cup \{i\}$ ▷ Keep unit in the network
      else
        Find $\{\alpha_k\}_{k \in S}$ such that $w_i^l = \sum_{k \in S} \alpha_k w_k^l$
        for $j \leftarrow 1, \ldots, n_{l+1}$ **do**
          for $k \in S$ **do**
            $w_{jk}^{l+1} \leftarrow w_{jk}^{l+1} + \alpha_k w_{ji}^{l+1}$
          end for
          $b_j^{l+1} \leftarrow b_j^{l+1} + w_{ji}^{l+1} \left(b_i^l - \sum_{k \in S} \alpha_k b_k^l\right)$
        end for
        Remove unit i from layer l ▷ Unit i is no longer necessary
      end if
    else
      Unstable ← True
    end if
  end for
  if not Unstable **then** ▷ All units left in layer l are stable
    if $|S| > 0$ **then** ▷ The units left have varying outputs
      Create matrix $\bar{\mathbf{W}} \in \mathbb{R}^{n_l \times n_{l+1}}$ and vector $\bar{\mathbf{b}} \in \mathbb{R}^{n_{l+1}}$
      for $i \leftarrow 1, \ldots, n_{l+1}$ **do**
        $\bar{b}_i \leftarrow b_i^{l+1} + \sum_{k \in S} w_{ik}^{l+1} b_k^l$
        for $j \leftarrow 1, \ldots, n_{l-1}$ **do**
          $\bar{w}_{ij} \leftarrow \sum_{k \in S} w_{kj}^l w_{ik}^{l+1}$
        end for
      end for
      Remove layer l; replace parameters in next layer with $\bar{\mathbf{W}}$ and $\bar{\mathbf{b}}$
    else ▷ Only unit left in layer l has constant output
      Compute output Υ for any input $\chi \in \mathbb{D}$
      $(W^{L+1}, b^{L+1}) \leftarrow (0, \Upsilon)$ ▷ Set constant values in output layer
      Remove layers 1 to L and **break** ▷ Remove all hidden layers and leave
    end if
  end if
end for

Training the models – variations

Trained with Pytorch, and used the MSE loss function.

Optimizers:

1. Adam
2. AdaDelta
3. SGD
4. SGD with momentum

Model architecture $[2,1024,512,512,256,1]$

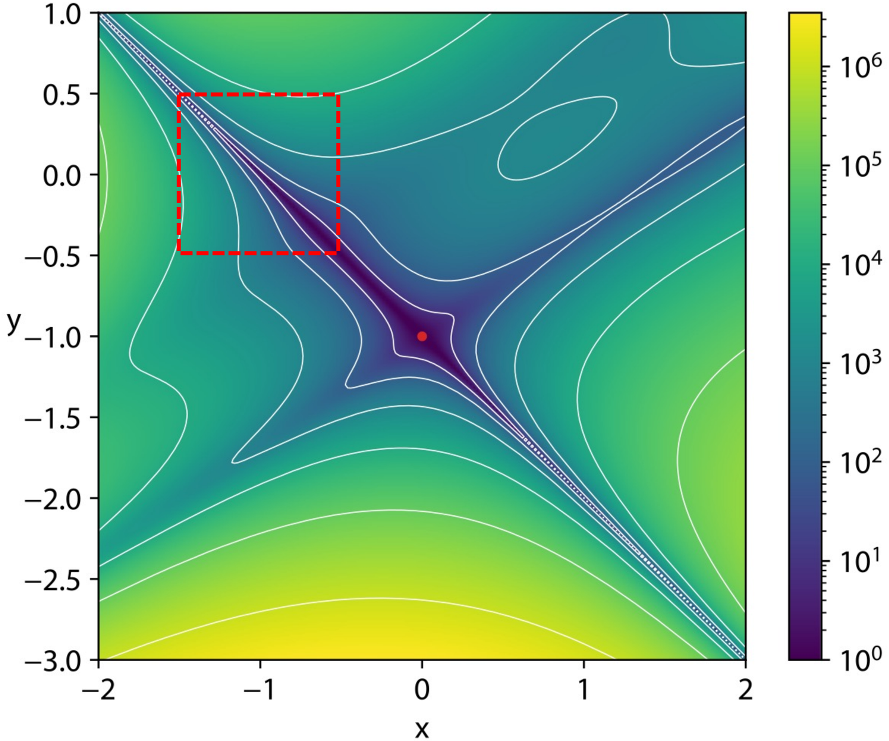
Regularization parameters:

1. L1 (Lasso) regularization, $\lambda_{l_1} \in [0,0.001,0.01,0.1,0.5]$
2. L2 (Ridge) regularization, $\lambda_{l_2} \in [0,0.001,0.01,0.1,0.5]$
3. L1+L2 (Elastic net) regularization, $\left(\lambda_{l_1}, \lambda_{l_2}\right) \in [(\lambda, \lambda), \lambda \in [0.001,0.01,0.1,0.5]$

→ $13 \times 4 = 52$ models

Data - Goldstein price

$$x \in [-1.5, -0.5]$$
$$y \in [-0.5, 0.5]$$

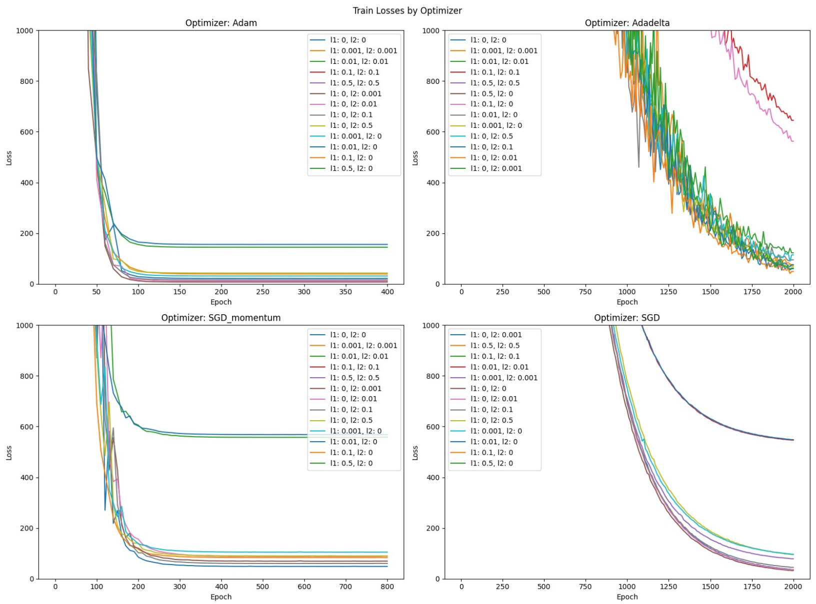


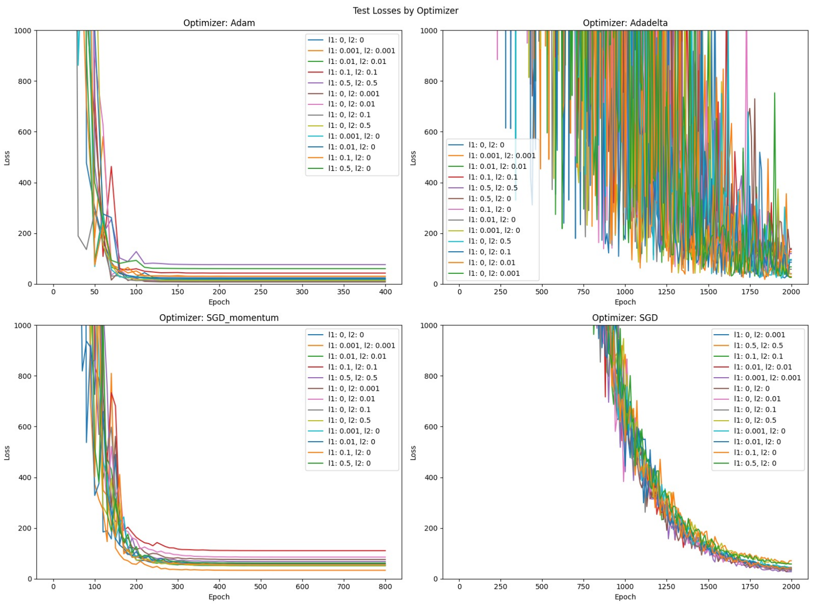
Training samples: 40,000
Testing samples: 8,000

Training the models – hyperparameters

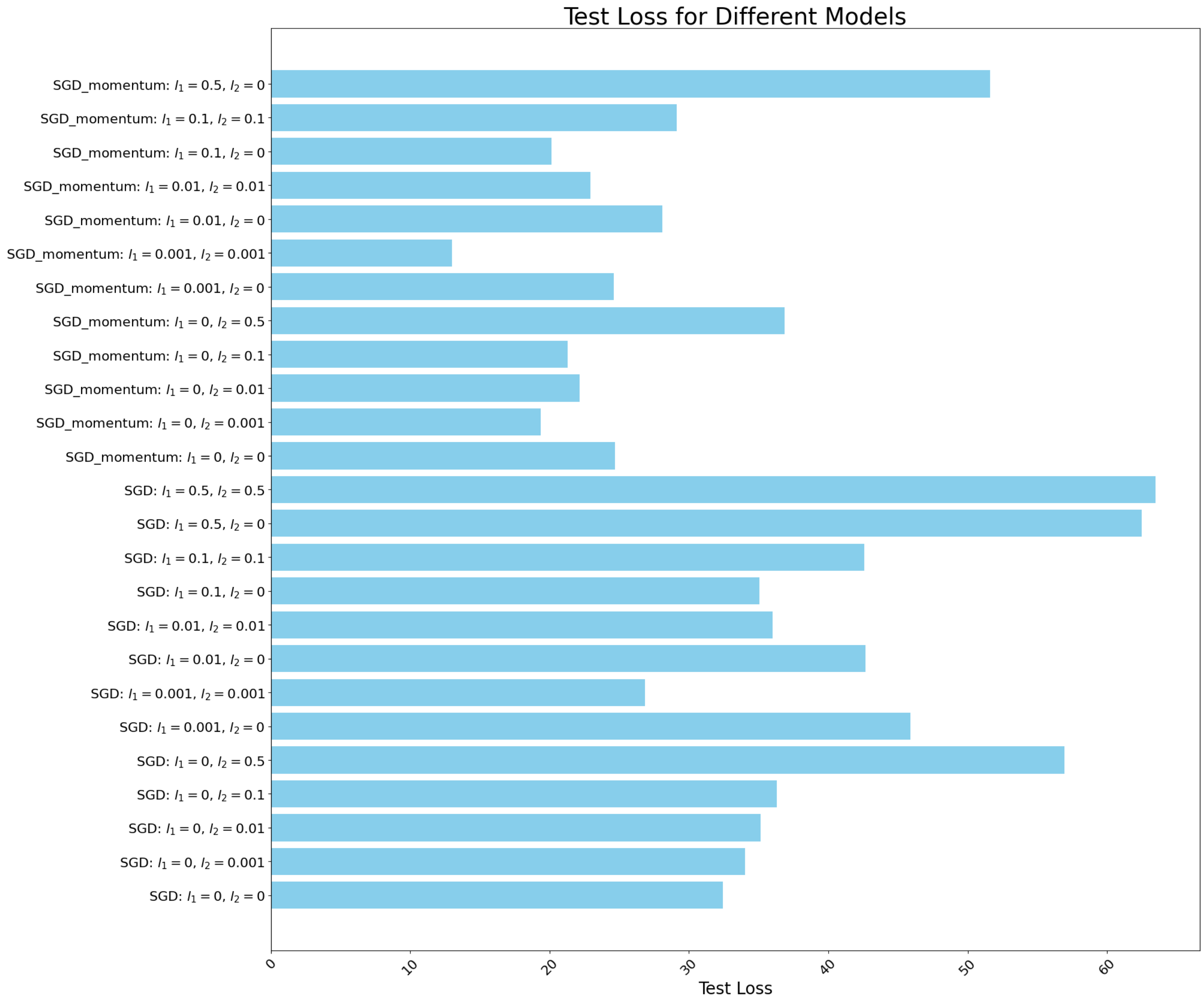
Optimizer	Epochs	Training time min/model on T4	Training time min/model on M2 Max	Learning rate	Learning rate decay γ	Gradient norm clipped to
Adam	600	10	14	0.001	0.95	5
AdaDelta	2000	31	45	1.5	0.9985	False
SGD	2000	30	43	0.025	0.998	5
SGD with momentum	800	13	19	0.005	0.995	5

Training the models – losses

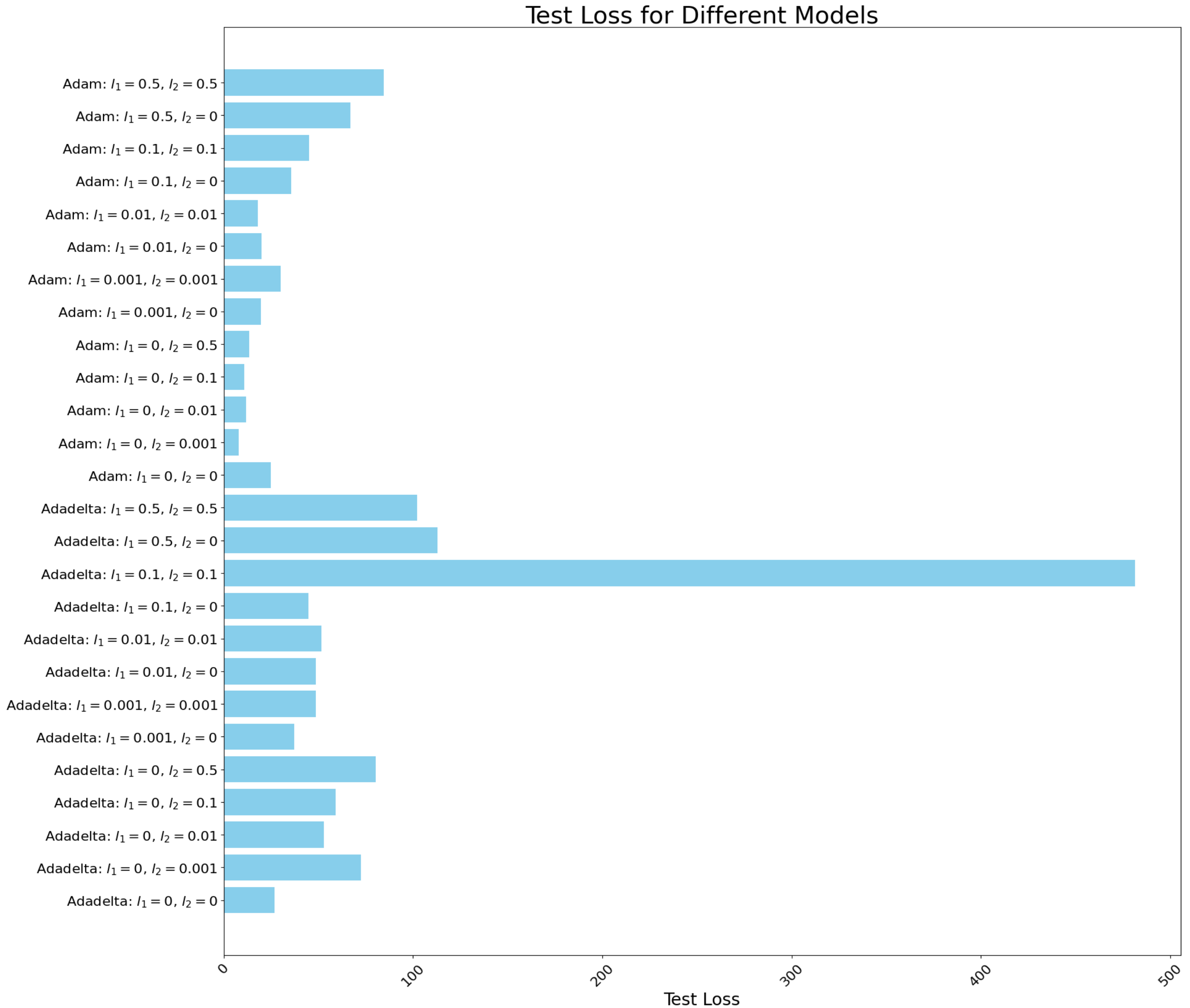




Testing the models



Testing the models

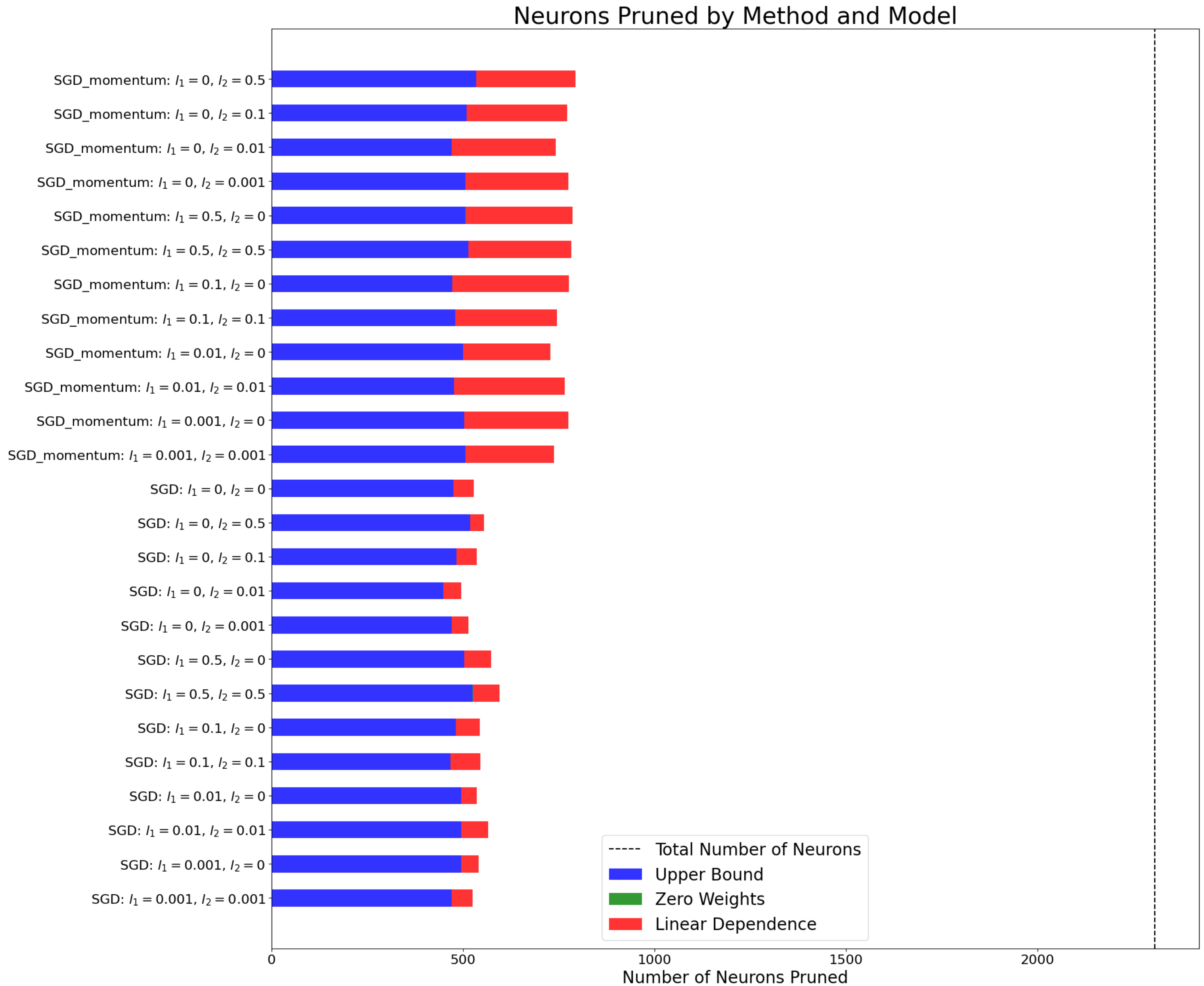


Bounding and pruning the network

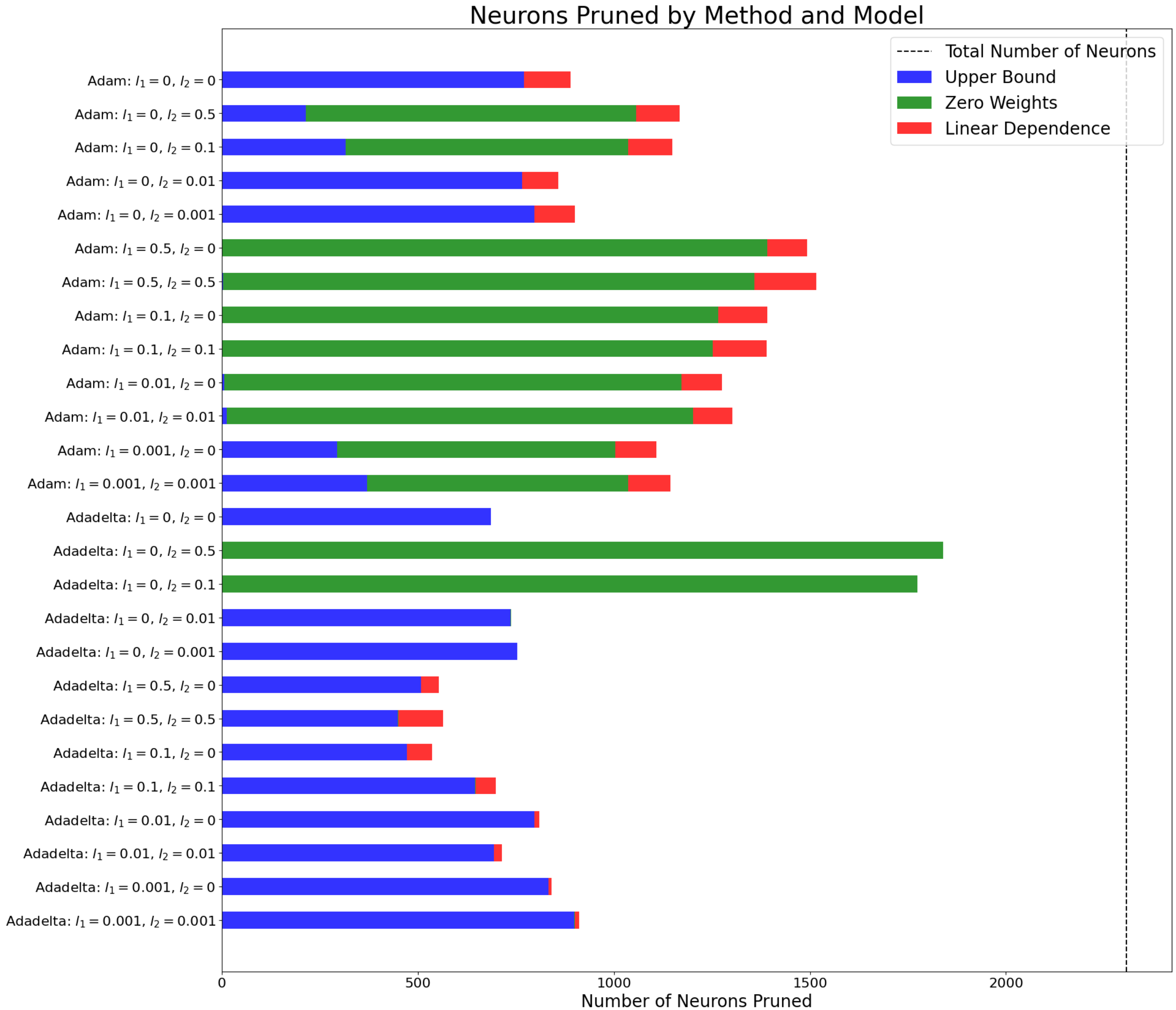
Bounds were calculated by computing the LP relaxations for the MIP problems for each neuron.

For each model, the bounding took ≈ 8 min.

Results


Neurons Pruned by Method and Model
SGD_momentum: $l_1 = 0$, $l_2 = 0.5$
SGD_momentum: $l_1 = 0$, $l_2 = 0.1$
SGD_momentum: $l_1 = 0$, $l_2 = 0.01$
SGD_momentum: $l_1 = 0$, $l_2 = 0.001$
SGD_momentum: $l_1 = 0.5$, $l_2 = 0$
SGD_momentum: $l_1 = 0.5$, $l_2 = 0.5$
SGD_momentum: $l_1 = 0.1$, $l_2 = 0$
SGD_momentum: $l_1 = 0.1$, $l_2 = 0.1$
SGD_momentum: $l_1 = 0.01$, $l_2 = 0$
SGD_momentum: $l_1 = 0.01$, $l_2 = 0.01$
SGD_momentum: $l_1 = 0.001$, $l_2 = 0$
SGD_momentum: $l_1 = 0.001$, $l_2 = 0.001$
SGD: $l_1 = 0$, $l_2 = 0$
SGD: $l_1 = 0$, $l_2 = 0.5$
SGD: $l_1 = 0$, $l_2 = 0.1$
SGD: $l_1 = 0$, $l_2 = 0.01$
SGD: $l_1 = 0$, $l_2 = 0.001$
SGD: $l_1 = 0.5$, $l_2 = 0$
SGD: $l_1 = 0.5$, $l_2 = 0.5$
SGD: $l_1 = 0.1$, $l_2 = 0$
SGD: $l_1 = 0.1$, $l_2 = 0.1$
SGD: $l_1 = 0.01$, $l_2 = 0$
SGD: $l_1 = 0.01$, $l_2 = 0.01$
SGD: $l_1 = 0.001$, $l_2 = 0$
SGD: $l_1 = 0.001$, $l_2 = 0.001$
Total Number of Neurons
Upper Bound
Zero Weights
Linear Dependence
0
500
1000
1500
2000
Number of Neurons Pruned

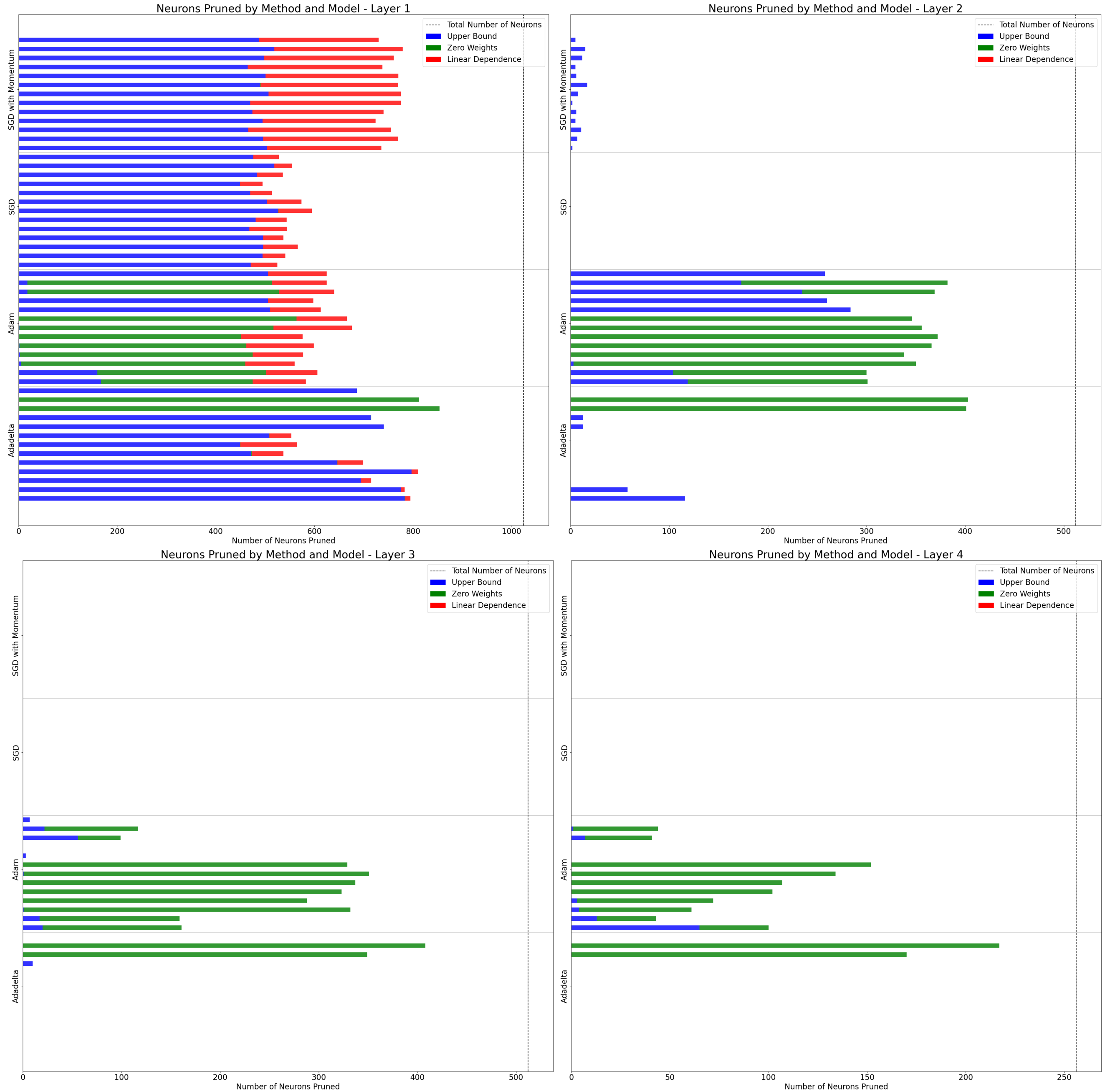
Results



Aalto-yliopisto
Perustieteiden
korkeakoulu

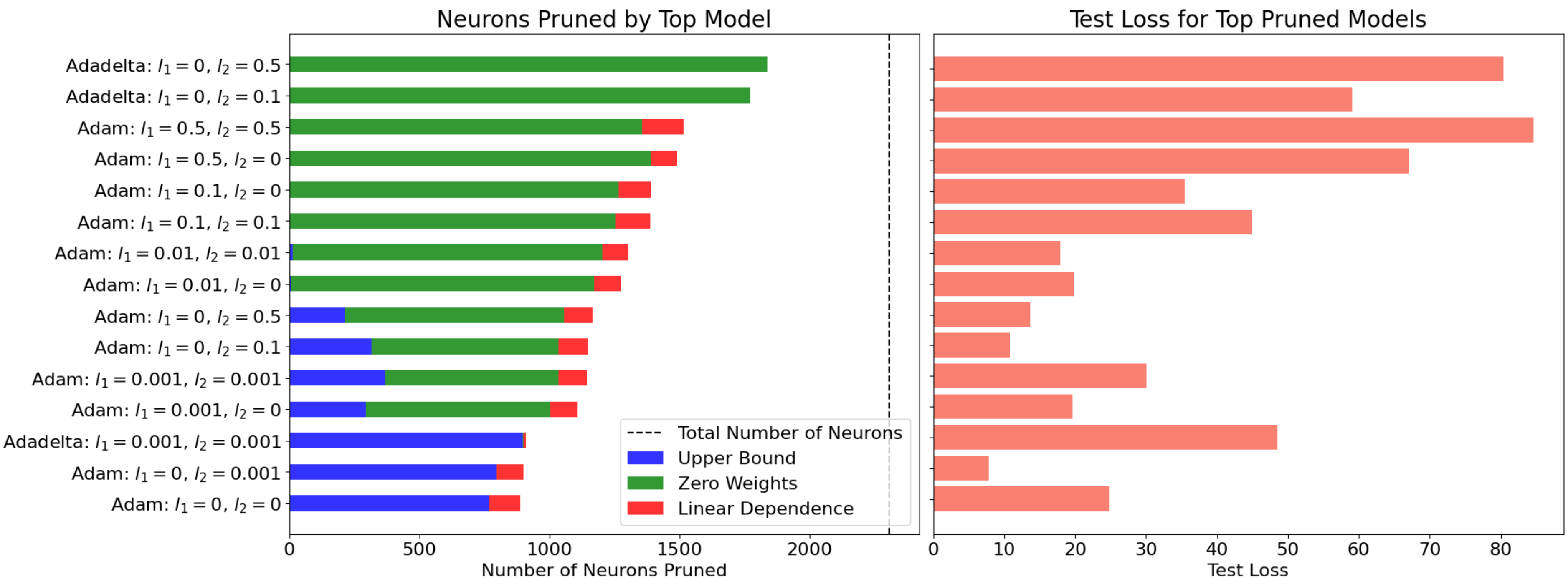
Systeemianalyysin
laboratorio

Results by layer



Aalto-yliopisto
Perustieteiden
korkeakoulu

Conclusions



The AdaDelta optimizer with $\lambda_{l_2} = 0.5$ or $\lambda_{l_2} = 0.1$ yield the best results.

The Adam optimizer yield consistently good results.

Aalto-yliopisto
Perustieteiden korkeakoulu

Systeemianalyysin laboratorio

Limitations

Only trained one model for each combination.

Only tried LP relaxed bounds.

Only considered strict inequalities in upper and lower bound clauses.

Only considered DNNs with the ReLU activation.

Did not consider other architectures, like CNNs, RNNs or attention models.

Sources and materials

- Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam 2020. Lossless Compression of Deep Neural Networks. Bucknell University and the University of Utah. Usa
- Linkola, J. 2023. Reformulating deep neural networks as mathematical programming problems. Bachelor thesis. Aalto-University. School of Science. Espoo.
- ML_as_MO package (https://github.com/gamma-opt/ML_as_MO). Includes implementations for calculating bounds