

A THE FULL ELF ALGORITHM

Algorithm 3 The ELF algorithm

Input: window_size (default: 150)

Input: steps_to_train

Input: loss_improvement_factor (default: 0.01)

```

1: losses  $\leftarrow []$ 
2: last_mean_loss  $\leftarrow 0$ 
3: t_of_last_update  $\leftarrow -1$ 
4: expected_per_step_improvement  $\leftarrow \infty$ 
5: t  $\leftarrow 0$ 
6: update_step  $\leftarrow 0$ 
7: while  $t < \text{steps\_to\_train}$  do
8:   real_improvement  $\leftarrow \text{last\_mean\_loss} - \text{mean}(\text{losses})$ 
9:   expected_improvement  $\leftarrow \text{last\_mean\_loss} -$ 
     expected_per_step_improvement  $\cdot (t - \text{t\_of\_last\_update})$ 
10:  if  $(t - \text{t\_of\_last\_update} + 1) \bmod (\text{window\_size} + 1) == 0$  and
     real_improvement  $\leq$  expected_improvement  $\cdot \text{loss\_improvement\_factor}$  then
11:    suggested_update_step, expected_per_step_improvement,
    num_batches_loaded_for_line_search  $\leftarrow$  perform_elf_line_search() {Algorithm 1}
12:    if suggested_update_step  $> 0$  then
13:      update_step  $\leftarrow$  suggested_update_step
14:    end if
15:    last_mean_loss  $\leftarrow \text{mean}(\text{losses})$ 
16:    losses  $\leftarrow []$ 
17:     $t \leftarrow t + \text{num\_batches\_loaded\_for\_line\_search}$ 
18:    t_of_last_update  $\leftarrow t$ 
19:  else
20:    loss  $\leftarrow$  perform_SGD_training_step(update_step)
21:    losses.append(loss)
22:     $t \leftarrow t + 1$ 
23:  end if
24: end while

```

B FURTHER EXPERIMENTAL DETAILS

For the experiments in Section 3.1 the following non default hyperparameters were considered:

GOLS1: initial step size: 0.001, 0.01, 0.1, 1

PLS: initial step size = 0.001, 0.01, 0.1, 1

ELF: momentum = 0.0, 0.4, decrease factor = 0, 0.2

The training steps for the experiments in section Section 3.2 were on all datasets except of ImageNet 100000 steps for DenseNet and 150000 steps for MobileNetv2, ResNet-18 and ResNet-34. On ImageNet 1687500 steps were trained. The batch size was 128 for all experiments, except for ImageNet where it was 90. The validation/train set splits were: 5000/45000 for CIFAR-10 and CIFAR-100 15000/60000 for Fashion MNIST 8257/65000 for SVHN 12000/1080000 for ImageNet.

All images were normalized with a mean and standard deviation determined over the dataset. On CIFAR-10, CIFAR-100 and SVHN we used random horizontal flips and random cropping with size 32 and with a padding of 8 for CIFAR-100 and of 4 for SVHN and CIFAR-10. On ImageNet we used random horizontal flips and and random resized crops with size 224.

To be compatible with the PLS implementation, Tensorflow 1.3 was used for the experiments in Section 3.1. Pytorch 1.6 was used for all other experiments.

C TRAINING LOSS AND VALIDATION ACCURACY PLOTS OF THE EXPERIMENTS IN SECTION 3.2

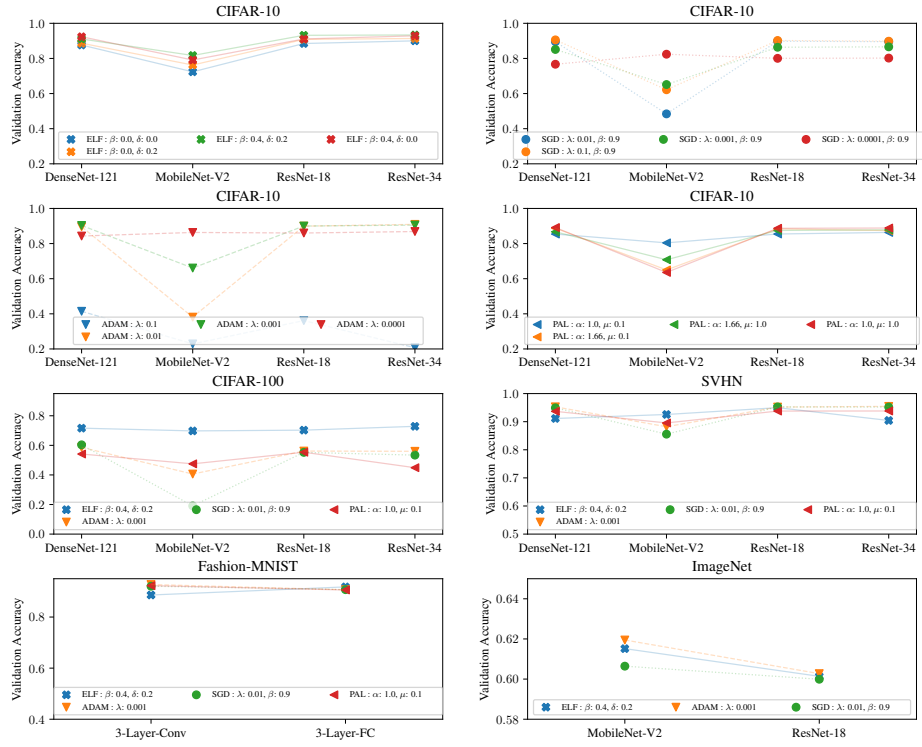


Figure 7: Corresponding Validation Accuracies to Figure 4. (β =momentum, λ =learning rate, δ =decrease factor)

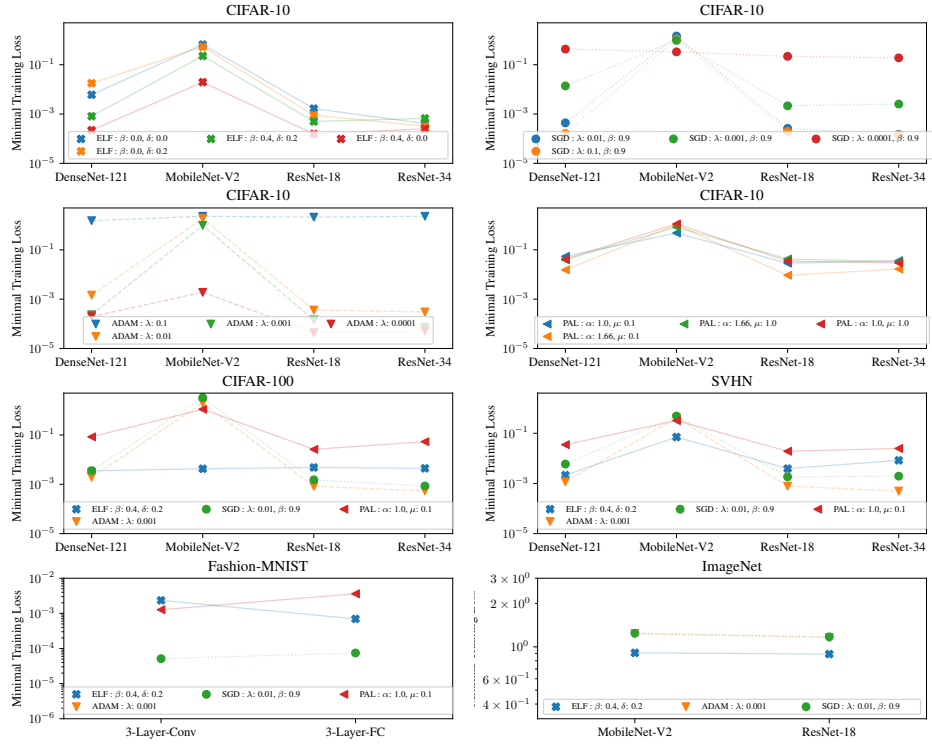


Figure 8: Corresponding training losses to Figure 4. (β =momentum, λ =learning rate, δ =decrease factor)

D FURTHER POLYNOMIAL LINE APPROXIMATIONS

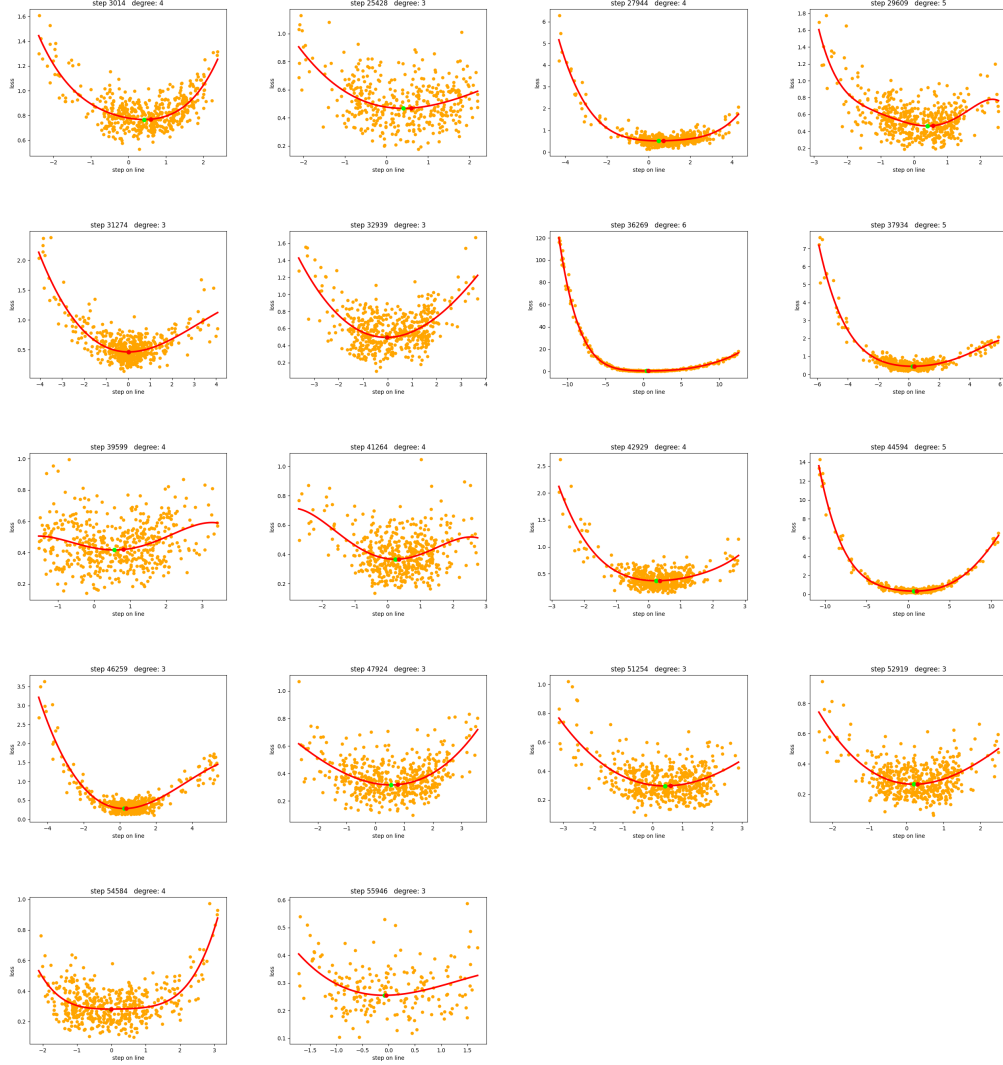


Figure 9: Polynomial line approximations (red) from a training of a DenseNet on CIFAR-10. The samples losses are orange. The minimum of the approximation is the green dot. The update step adjusted by a decrease factor of 0.2 is the red dot.

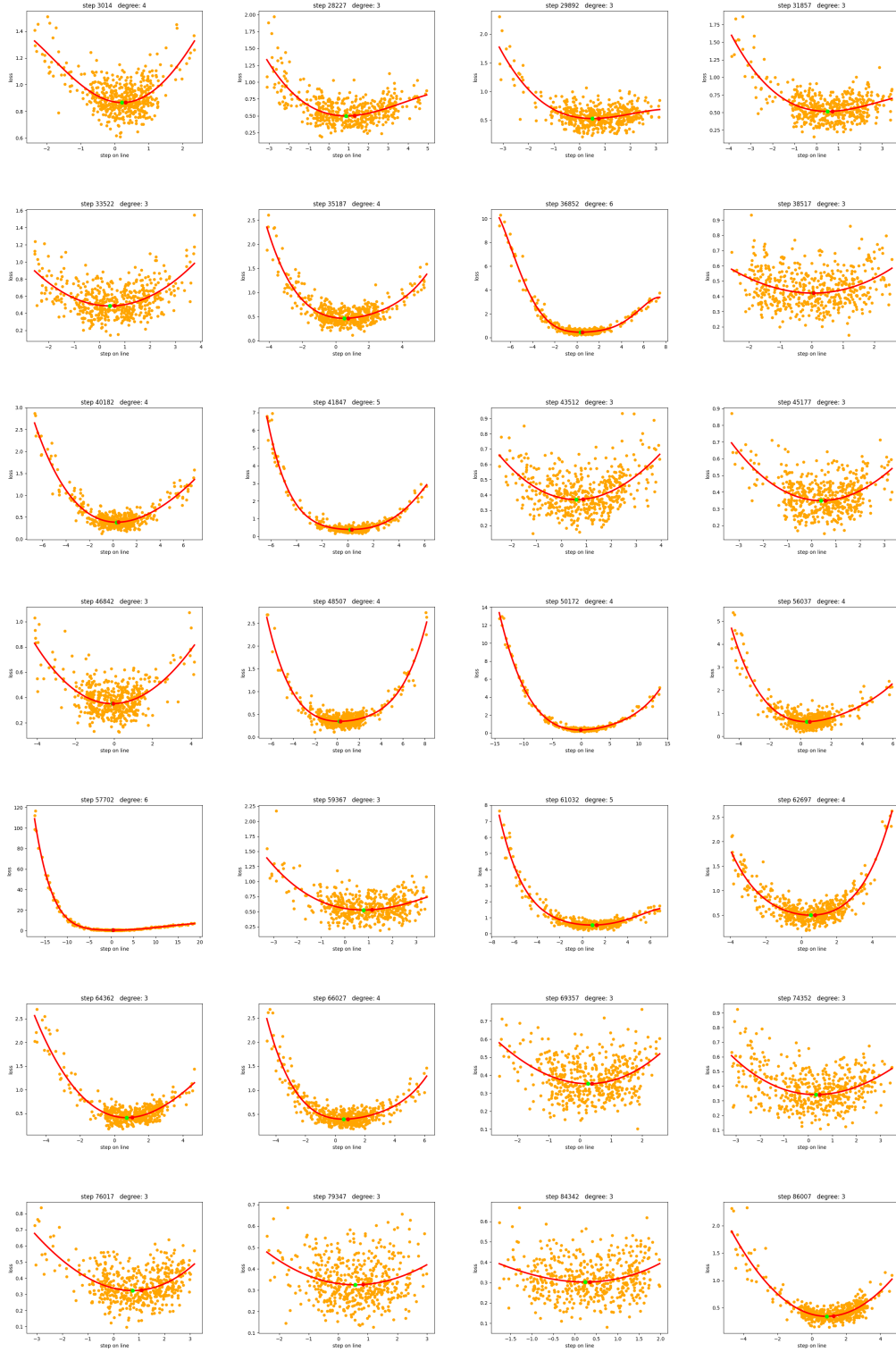


Figure 10: Polynomial line approximations (red) from a training of a ResNet18 on CIFAR-10. The samples losses are orange. The minimum of the approximation is the green dot. The update step adjusted by a decrease factor of 0.2 is the red dot.