Channel selection using Gumbel Softmax: Supplement

Charles Herrmann[1], Richard Strong Bowen[1], and Ramin Zabih[1,2]

[1] Cornell Tech
{cih, rsb, rdz}@cs.cornell.edu
[2] Google Research

1 Implementation details

1.1 Granularity

For faster computation, the results in the paper group several filters together under a single gate. We use the term "granularity" to refer to the number of filters that a single gate controls. The higher the granularity, the closer we move to one gate per filter. We were noticed that lower granularities have better early performance. For the numbers reported in the paper, we use the below settings, which try to keep the number of flops gated by a single gate roughly constant. In general, we tried to roughly equalize the number of FLOPs. For non-residual layers, we also force one gate open.

t	c	n	s	g_{inp} granularity	g_{out} granularity	g_{exp} granularity
1	16	1	1	8	8	16
6	24	2	2	8	8	24
6	32	3	2	8	16	24
6	64	4	2	8	16	32
6	96	3	1	16	32	48
6	160	3	2	16	32	64
6	320	1	1	32	64	64

For ResNet-50, each gate controls 16, 32, 64 or 128 filters in the four layers' expansions, and half that in the contractions.

1.2 Additional Early Exit Details

Here, we focus on the use of probabilistic gates to provide an early exit, when the network is sufficiently certain of the answer. We are motivated by MSDNet [1], which investigated any-time classification. We explored early exit on both ResNet and DenseNet; however, consistent with [1], we found that ResNet tended to degrade with intermediate classifiers while DenseNet did not.

Probabilistic gates can be used for early exit in DenseNet; following [5] we place gates and intermediate classifiers at the end of each dense block. At each gate, the network makes a discrete decision as to whether the instance can be successfully classified at that stage. The advantage of using Gumbel here is that the early exit can be trained in an end-to-end fashion unlike [5] which uses reinforcement learning.

These early exit gates can make good decisions regarding which instances to classify early: the intermediate classifiers have higher accuracy on the instances chosen by the gates. Results are shown in DenseNet figure in the paper.

The network had an overall accuracy of 94.39 while use on average only 68.4% of the layers; our implementation of the original DenseNet architecture achieves an accuracy of 95.24 ([3] reports 95.49). More than a third of examples exited early, while overall error was still low.

For our early exit classifiers, we use the same classifiers as [1]. For the gate structure, we use a stronger version of the gate described by [6]. The gates are comprised of the following: a 3×3 convolutional layer with stride of 1 and padding of 1 which takes the current state of the model and outputs 128 channels, a BatchNorm, another 3×3 convolutional layer with stride of 1 and padding of 1 which outputs 128 channels, a BatchNorm, a 4×4 average pool, a linear layer, and then finally a GumbleSoftmax.

2 Observed Polarization

We observe a strong tendency towards gate probability polarization. For example, in a typical run, a histogram of the learned gate probabilities is included in Figure 1.

3 Additional data and experiments with inference techniques

3.1 Results for different inference techniques

We report the measured FLOPs and accuracy for several inference styles in Figure 1.

	($\tau = 0.5$)		($\tau = 0.8$)			
Model	mFLOPs	Top1	mFLOPs	Top1	Top1 ensem	Top1 stoch
Resnet-50 Cond $t = .5$	2.21	76.30	2095.71	75.86	76.39	75.75
Resnet-50 Cond $t = .4$	1.94	76.04	1836.57	75.57	76.09	75.43
Resnet-50 Cond $t = .3$	1.67	75.19	1587.61	74.82	75.54	74.82
Resnet-50 Prune $t = .5$	2.51	76.20	2494.19	76.25	76.15	75.92
Resnet-50 Prune $t = .4$	2.28	76.14	2264.99	76.12	76.13	75.80
Resnet-50 Prune $t = .3$	1.97	75.56	1935.83	75.52	75.50	75.17

Table 1: Results on Imagenet for a number of inference styles, including thresholding at two values, an ensemble of 5 different stochastic runs, stochastic (reported accuracies are a mean of 5 runs)

3.2 Detailed results for difference inference techniques on ResNet-50

Table 1 includes the results of running all inference techniques on the ResNet-50 data for the models reported in the paper (using the filter-based granularity). The relation between the inference techniques is consistent across models.

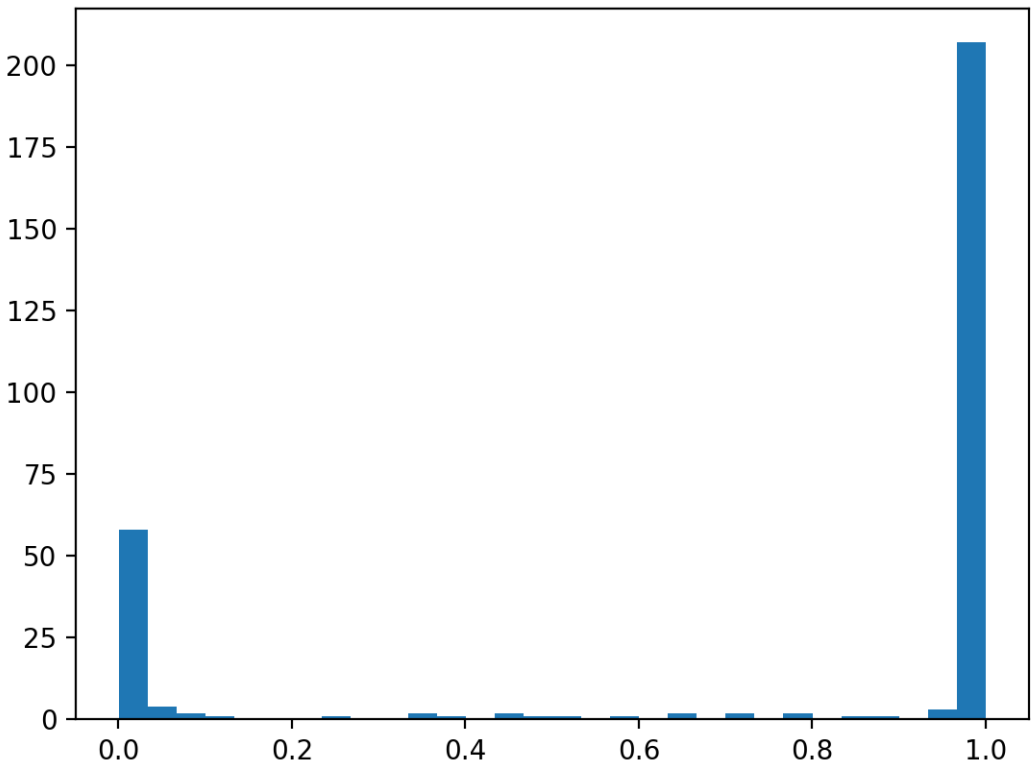

Fig. 1: A histogram of learned gate probabilities for a typical pruning run. They demonstrate strong polarization. The x-axis is gate activation rate. The y-axis is the number of gates with that gate activation rate.

4 Layer-based Approach

4.1 Approach description

We also investigated a layer-pruning approach for ResNet. As in AIG, for a residual layer $f_l()$, we typically have

$$x_{l+1} = x_l + f_l(x_l)$$

which we simply replace by a probabilistic gate to get

$$x_{l+1} = x_l + Z_l(f_l(x_l))$$

so that the layer's being run is dependent on the value of the gate. In the below, we report layer-pruning results; the baseline model is ResNet-50. For brevity we adopt the following abbreviations:

- Loss functions are either **PB** (batch activation loss, our proposed loss) or **PG** (per-gate, as in AIG).
- **Ind** and **Dep** stand for independent (pruning) or dependent (conditional computation) gates.
- **Act** measures the overall activations, i.e., the (average) fraction of the gates that are on

4.2 Polarization

The following is a summary of polarization on the layer-based granularity.

With the per-batch loss, we often observe polarization, where some layers are nearly always on and some nearly always off. In the case of data-dependent bypass, we can measure the observed activation of a gate during training. For example, on a per-batch run on ResNet-50 (16 gates) on ImageNet, nearly all of the 16 gates polarize, as shown in Figure 2: four gates collapsed to zero or one exactly; more than half were at their mode more than 99.9% of the time. Interestingly, we observe different activation behavior on different datasets. ImageNet leads to frequent and aggressive polarization, all networks exhibited some degree of polarization; CIFAR10 can induce polarization but does so much less frequently, approximately less than 40% of our runs.

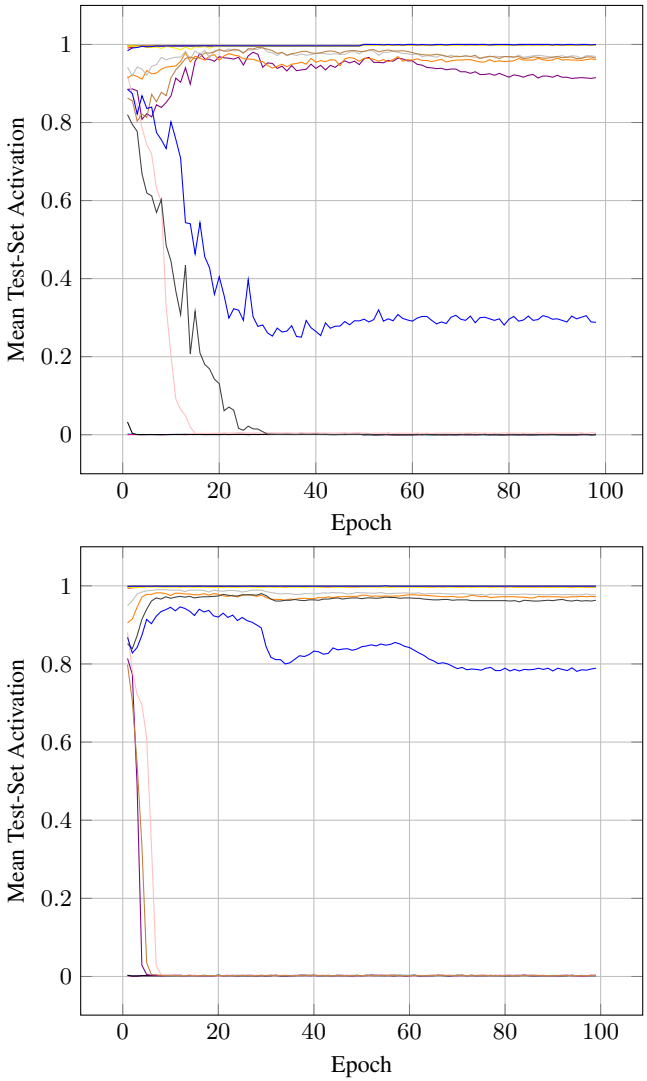

Fig. 2: Demonstration of polarization on layer-based granularity on (left) data-dependent, per-batch ResNet-50 on ImageNet with target rate of .5, and (right) data-independent per-batch with target rate of .4 (right). Nearly all of the 16 gates collapse.

4.3 ImageNet Results

In Figure 4 and Tables 5, 6, 7 and 8 we report all data collected on a layer-based granularity. See Figure 3 for a summary table.

We report all the data collected on ImageNet using the different gate techniques (independent, dependent), target loss techniques (per-batch, per-gate), and inference time

Model	Top-1 Stochastic	Top-5 Stochastic	GFLOPs Stochastic	Top-1 Det.	Top-5 Det.	# GFLOPs Det.
ResNet-34	-	-	-	26.69	8.58	3.6
ResNet-50	-	-	-	23.87	7.12	3.8
AIG 50 [t=0.4]	24.75	7.61	2.56	-	-	2.56
AIG 50 [t=0.5]	24.42	7.42	2.71	-	-	2.71
AIG 50 [t=0.6]	24.22	7.21	2.88	-	-	2.88
AIG 50 [t=0.7]	23.82	7.08	3.06	-	-	3.06
Ind PG* [t=0.5]	24.99 (0.05)	7.71 (0.04)	3.81	24.23	7.17	3.04
Dep PG* [t=0.4]	25.25 (0.08)	7.81 (0.05)	2.51	24.78	7.57	2.52
Dep PG* [t=0.5]	24.92 (0.05)	7.50 (0.02)	2.73	24.52	7.27	2.79
Dep PG* [t=0.6]	24.47 (0.07)	7.36 (0.05)	2.99	24.07	7.16	3.03
Ind PB* [t=0.4]	24.70 (0.08)	7.63 (0.03)	2.43	24.42	7.48	2.49
Ind PB* [t=0.5]	24.39 (0.05)	7.46 (0.03)	2.65	24.04	7.29	2.71
Ind PB* [t=0.6]	**24.04 (0.03)**	7.11 (0.03)	2.93	**23.72**	**6.93**	2.93
Dep PB* [t=0.4]	24.98 (0.02)	7.63 (0.10)	2.27	24.75	7.56	2.28
Dep PB* [t=0.5]	24.22 (0.04)	7.18 (0.03)	2.52	23.99	7.06	2.55
Dep PB* [t=0.6]	24.16 (0.05)	7.24 (0.01)	2.71	23.99	7.14	2.73
ResNet-101	-	-	-	22.63	6.45	7.6
AIG 101 [t=0.3]	23.02	6.58	4.33	-	-	-
AIG 101 [t=0.4]	22.63	6.26	5.11	-	-	-
Dep PB* [t=0.5]	22.73 (0.02)	6.46 (0.02)	4.48	22.43	6.34	4.60
Dep PB* [t=0.6]	**22.45 (0.08)**	6.28 (0.04)	5.11	**22.22**	6.28	5.28

Fig. 3: Error and GFLOPs for our method (marked with asterisks) compared to ConvNet-AIG. For stochastic errors we run the test set 5 times and report mean and, in parenthesis, standard deviation. For deterministic error, we use the thresholding inference technique.

techniques (threshold, always-on, stochastic, ensemble). Note that we try to include AIG for reference whenever it's a fair comparison.

Also of note, for the ensemble technique we also include data from [2]. Note that using the stochastic networks, we outperform their ensemble technique. Also note that their technique is orthogonal to ours, so both could be used to identify an even better ensemble.

In general, we observe that unsurprisingly, ensemble has the highest performance in terms of error; however, this requires multiple forward passes through the network, so the performance gain is offset by the inference time required. We also observe that threshold generally outperforms stochastic.

For all ImageNet results, we used the pretrained models provided by TorchVision[3].

[3] The model links are provided here: https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

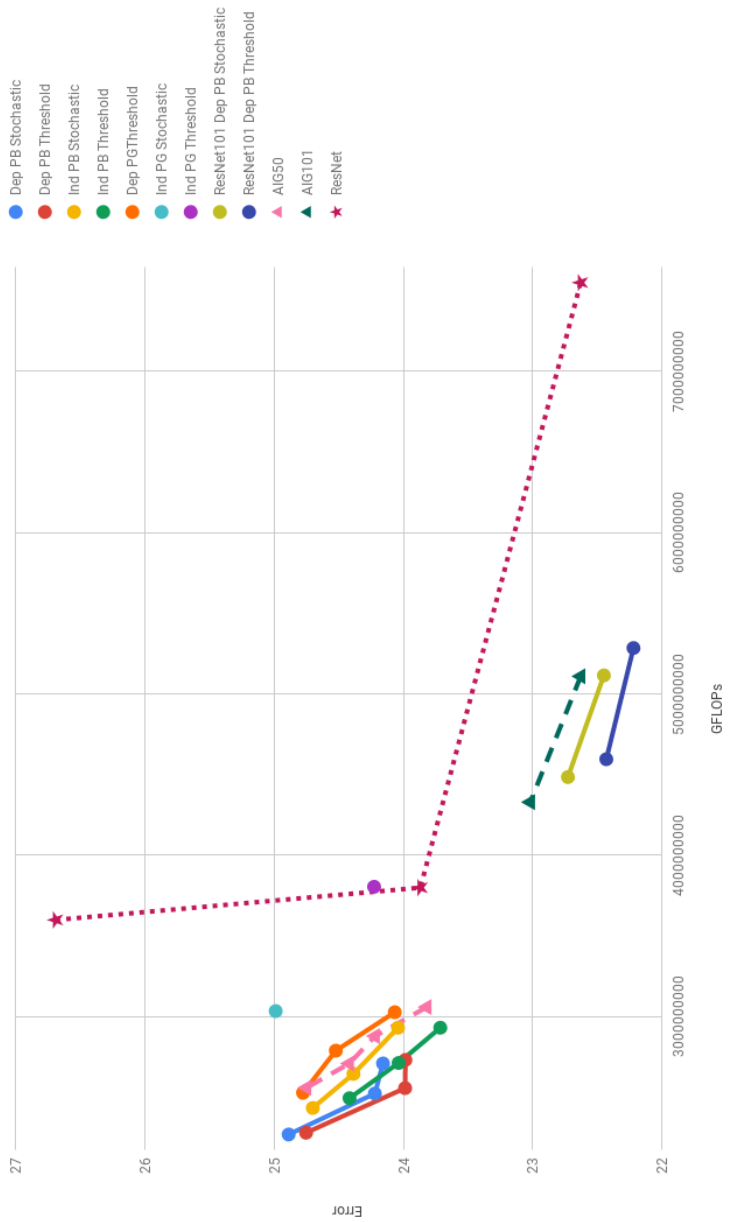

Fig. 4: Full comparison of the tradeoff for inference techniques for ResNet. Note the y-axis reports Prec@1 Error.

Technique	Threshold GFlops	Threshold Prec@1	Threshold Prec@5	Threshold Acts
Dep PB @ tr=.6	2732360786	23.986	7.14	0.6929
Dep PB @ tr=.5	2557646188	23.988	7.058	0.6429
Dep PB @ tr=.4	2281151219	24.754	7.562	0.5637
Ind PB @ tr=.6	2932012776	**23.716**	**6.934**	0.75
Ind PB @ tr=.5	2713646824	24.038	7.294	0.6875
Ind PB @ tr=.4	2495280872	24.418	7.478	0.625
Dep PG @ tr=.6	3027899835	24.068	7.16	0.7726
Dep PG @ tr=.5	2789914344	24.524	7.272	0.69
Dep PG @ tr=.4	2528475066	24.778	7.57	0.6098
Ind PG @ tr=.5	3805476584	24.228	7.168	1
AIG @ tr=.7	3060000000	23.82	7.08	-
AIG @ tr=.6	2880000000	24.22	7.21	-
AIG @ tr=.5	2710000000	24.42	7.41	-
AIG @ tr=.4	2560000000	24.75	7.61	-
Res101 Dep PB @ tr=.6	5284996031	**22.222**	6.28	0.69
Res101 Dep PB @ tr=.5	4596138799	22.432	6.336	0.594
AIG101 @ tr=.5	5110000000	22.62	**6.26**	-
AIG101 @ tr=.3	4330000000	23.02	6.58	-

Fig. 5: Full comparison of the tradeoff for inference techniques for ResNet (Thresholding). Note that this table reports Error for Prec@1 and Prec@5.

Technique	AlwaysOn Prec@1	AlwaysOn Prec@5	AlwaysOn Acts
Dep PB @ tr=.6	23.968	7.134	1
Dep PB @ tr=.5	24.362	7.248	1
Dep PB @ tr=.4	25.25	7.826	1
Ind PB @ tr=.6	**23.75**	**6.944**	1
Ind PB @ tr=.5	24.08	7.362	1
Ind PB @ tr=.4	24.46	7.448	1
Dep PG @ tr=.6	23.864	6.968	1
Dep PG @ tr=.5	24.406	7.19	1
Dep PG @ tr=.4	24.734	7.536	1
Ind PG @ tr=.5	24.228	7.168	1
ResNet 50	23.87	7.12	1
Res101 Dep PB @ tr=.6	**22.42**	**6.222**	1
Res101 Dep PB @ tr=.5	23.276	6.708	1
ResNet 101	22.63	6.45	

Fig. 6: Full comparison of the tradeoff for inference techniques for ResNet (Always-run-gates). Note that this table reports Error for Prec@1 and Prec@5.

Stochastic Technique	GFlops	Prec@1 Mean	Prec@1 StdDev	Prec@5 Mean	Prec@5 StdDev	Acts Mean	Acts StdDev
Dep PB @ tr=.6	2710541198	24.1592	0.047	7.24	0.0055	0.6862	0.0002
Dep PB @ tr=.5	2524273294	24.222	0.0358	7.18	0.0278	0.6321	0.0002
Dep PB @ tr=.4	2270347346	24.8892	0.0224	7.6284	0.096	0.5596	0.0001
Ind PB @ tr=.6	2932012776	24.0428	0.0246	7.1056	0.0282	0.7222	0.0001
Ind PB @ tr=.5	2646139684	24.3876	0.0496	7.46	0.0299	0.6676	0.0001
Ind PB @ tr=.4	2434652927	24.7024	0.0753	7.634	0.0291	0.6068	0.0002
Dep PG @ tr=.6	2988683827	24.472	0.0741	7.3616	0.0508	0.7549	0.0002
Dep PG @ tr=.5	2734859869	24.9176	0.052	7.4972	0.0231	0.6797	0.0002
Dep PG @ tr=.4	2510905649	25.2476	0.0802	7.8084	0.047	0.6136	0.0002
Ind PG @ tr=.5	3035128056	24.9892	0.05	7.7064	0.0345	0.7681	0.0005
AIG @ tr=.7	3060000000	23.82	-	7.08	-	-	-
AIG @ tr=.6	2880000000	24.22	-	7.21	-	-	-
AIG @ tr=.5	2710000000	24.42	-	7.41	-	-	-
AIG @ tr=.4	2560000000	24.75	-	7.61	-	-	-
Res101 Dep PB @ tr=.6	5115483937	**22.4508**	0.077	6.2768	0.0346	0.6665	0
Res101 Dep PB @ tr=.5	4485480286	22.7288	0.0229	6.4552	0.0206	0.5791	0.0001
AIG101 @ tr=.5	5110000000	22.62		**6.26**			
AIG101 @ tr=.3	4330000000	23.02		6.58			

Fig. 7: Full comparison of the tradeoff for inference techniques for ResNet (Stochastic). Note that this table reports Error for Prec@1 and Prec@5.

Technique	Ensemble Prec@1	Ensemble Prec@5
Dep PB @ tr=.6	23.952	7.14
Dep PB @ tr=.5	24.014	7.076
Dep PB @ tr=.4	24.76	7.558
Ind PB @ tr=.6	**23.814**	**6.97**
Ind PB @ tr=.5	24.116	7.32
Ind PB @ tr=.4	24.448	7.506
Dep PG @ tr=.6	23.844	7.1
Dep PG @ tr=.5	24.342	7.188
Dep PG @ tr=.4	24.62	7.486
Ind PG @ tr=.5	24.376	7.328
ResNet 50	23.87	7.12
Snapshot [2]	23.96[4]	
Res101 Dep PB @ tr=.6	**22.186**	**6.162**
Res101 Dep PB @ tr=.5	22.42	6.31
ResNet 101	22.63	6.45

Fig. 8: Full comparison of the tradeoff for inference techniques for ResNet (Ensembles). Note that this table reports Error for Prec@1 and Prec@5.

4.4 CIFAR10 Results

CIFAR10 Performance In Table 2 and Figure 9, we report all the data collected on CIFAR10 using the different gate techniques (independent, dependent), target loss techniques (per-batch, per-gate). We report only the numbers for the stochastic inference time technique. We used CIFAR10 as a faster way to explore the space of parameters and combinations and as such have a more dense sweep of the combination and parameters. Note that for CIFAR10, we did not use a pretrained model; the entire model is trained from scratch.

In general, we found that for a wide set of parameters per-batch outperforms per-gate. This includes independent per-batch outperforming dependent per-gate.

The only exception to this is very high and very low target rates. However we note that at very high target rates, the accuracy of per-batch can be recovered through annealing the target rate from high to low. We attribute this to the fact that for CIFAR10 we train from scratch. Since the model is completely blank for the first several epochs, the per-batch loss can lower activations for any layers while still improving accuracy. In other words, at the beginning, the model is so inaccurate that any training on any subset of the model will result in a gain of accuracy; so when training from scratch, the per-

batch loss will choose the layers to decrease activations for greedily and sub-optimally. However, many of these annealing techniques did not work on ImageNet.

One surprising result is that independent per-gate works at all for a wide range of target rates. This suggests that the redundancy effect described in [4] is so strong that the gates can be kept during inference time. This also suggests that at least for CIFAR10, most of the gains described in [6] were from regularization and not from specialization.

CIFAR10 Activation Graphs In Figures 10, 11, 12 and 13 we provide graphs of the activation rates for each layer over time. This demonstrates that on CIFAR10, each layer does not instantly polarize to its eventual activation rate; the activation rates can change throughout training time.

Technique	Best Error	Final Error	Activations
Dep PG @tr=.0	9.12	9.26	0.125
Dep PG @tr=.1	9.42	9.8	0.209
Dep PG @tr=.2	9.38	9.4	0.291
Dep PG @tr=.3	8.57	8.88	0.374
Dep PG @tr=.4	7.83	8.3	0.454
Dep PG @tr=.5	7.5	7.5	0.54
Dep PG @tr=.6	7	7.14	0.627
Dep PG @tr=.7	6.58	6.82	0.71
Dep PG @tr=.8	5.9	6.08	0.79
Dep PB @tr=.0	10.34	10.5	0.097
Dep PB @tr=.1	9.14	9.36	0.154
Dep PB @tr=.2	8.09	8.45	0.242
Dep PB @tr=.3	8.04	8.48	0.332
Dep PB @tr=.4	7.43	7.8	0.419
Dep PB @tr=.5	6.62	7.04	0.509
Dep PB @tr=.6	6.86	7.86	0.6
Dep PB @tr=.7	6.48	6.83	0.69
Dep PB @tr=.8	6.35	6.35	0.78
Ind PG @tr=.0	10.37	10.75	0.14
Ind PG @tr=.1	9.87	10.27	0.241
Ind PG @tr=.2	8.8	9.15	0.336
Ind PG @tr=.3	8.36	8.75	0.416
Ind PG @tr=.4	7.76	7.95	0.49
Ind PG @tr=.5	7.15	7.43	0.568
Ind PG @tr=.6	6.67	6.98	0.645
Ind PG @tr=.7	6.14	6.54	0.72
Ind PG @tr=.8	6.25	6.82	0.797
Ind PB @tr=.0	9.38	9.52	0.125
Ind PB @tr=.1	8.08	8.43	0.195
Ind PB @tr=.2	8.23	8.41	0.256
Ind PB @tr=.3	7.4	7.65	0.338
Ind PB @tr=.4	7.26	7.71	0.43
Ind PB @tr=.5	7.1	7.39	0.517
Ind PB @tr=.6	6.46	6.92	0.643
Ind PB @tr=.7	6.52	6.86	0.697
Ind PB @tr=.8	6.21	6.23	0.783

Table 2: Full list of CIFAR-10 results for ResNet-110 at various activations. Note that this table reports Error.

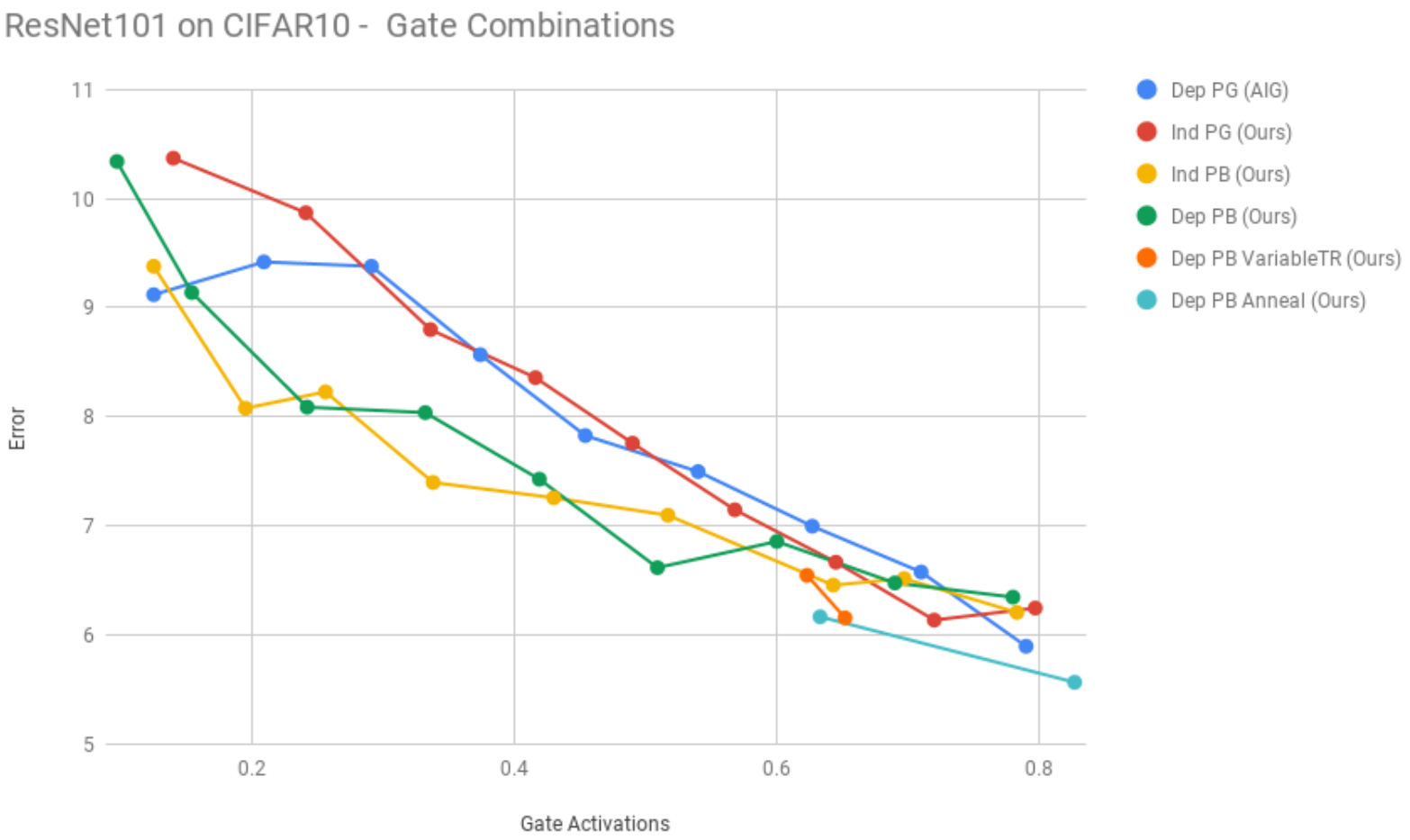

Fig. 9: Error rates of layer-based ResNet-110 with different target rates (gate activations).

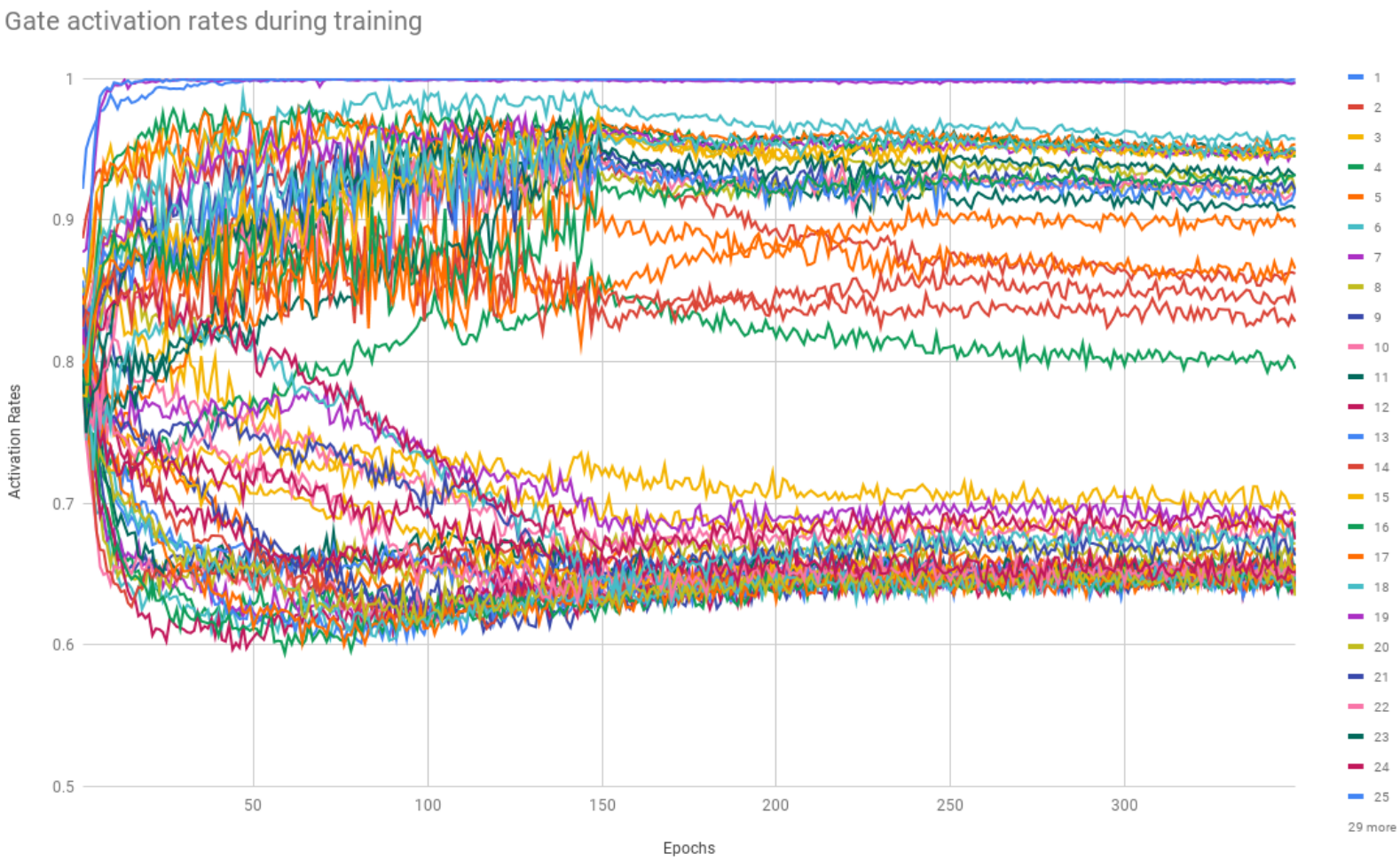

Fig. 10: Layer-based ResNet-110 PB with target rate of 0.8

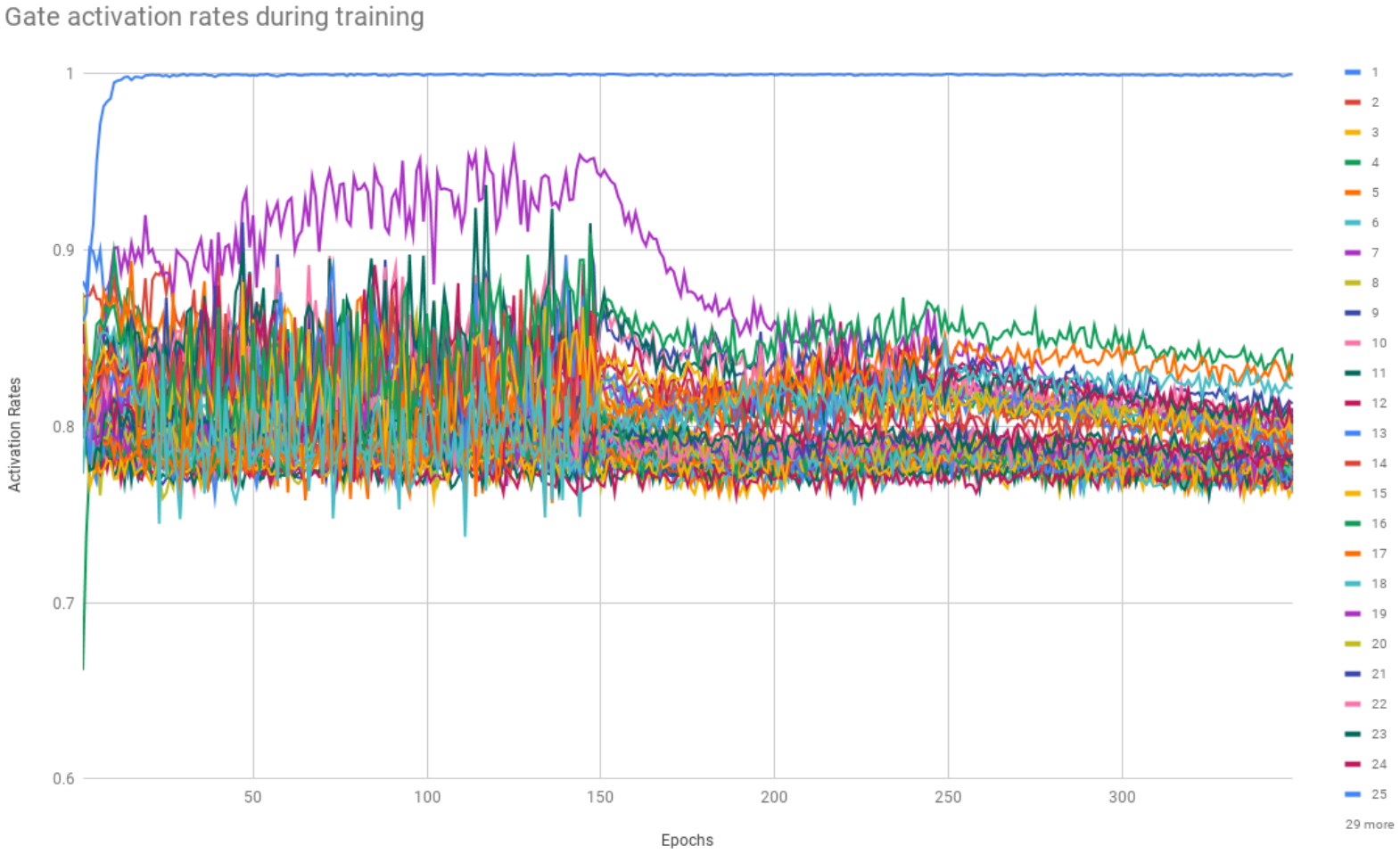

Fig. 11: Layer-based ResNet-110 PG with target rate of 0.8

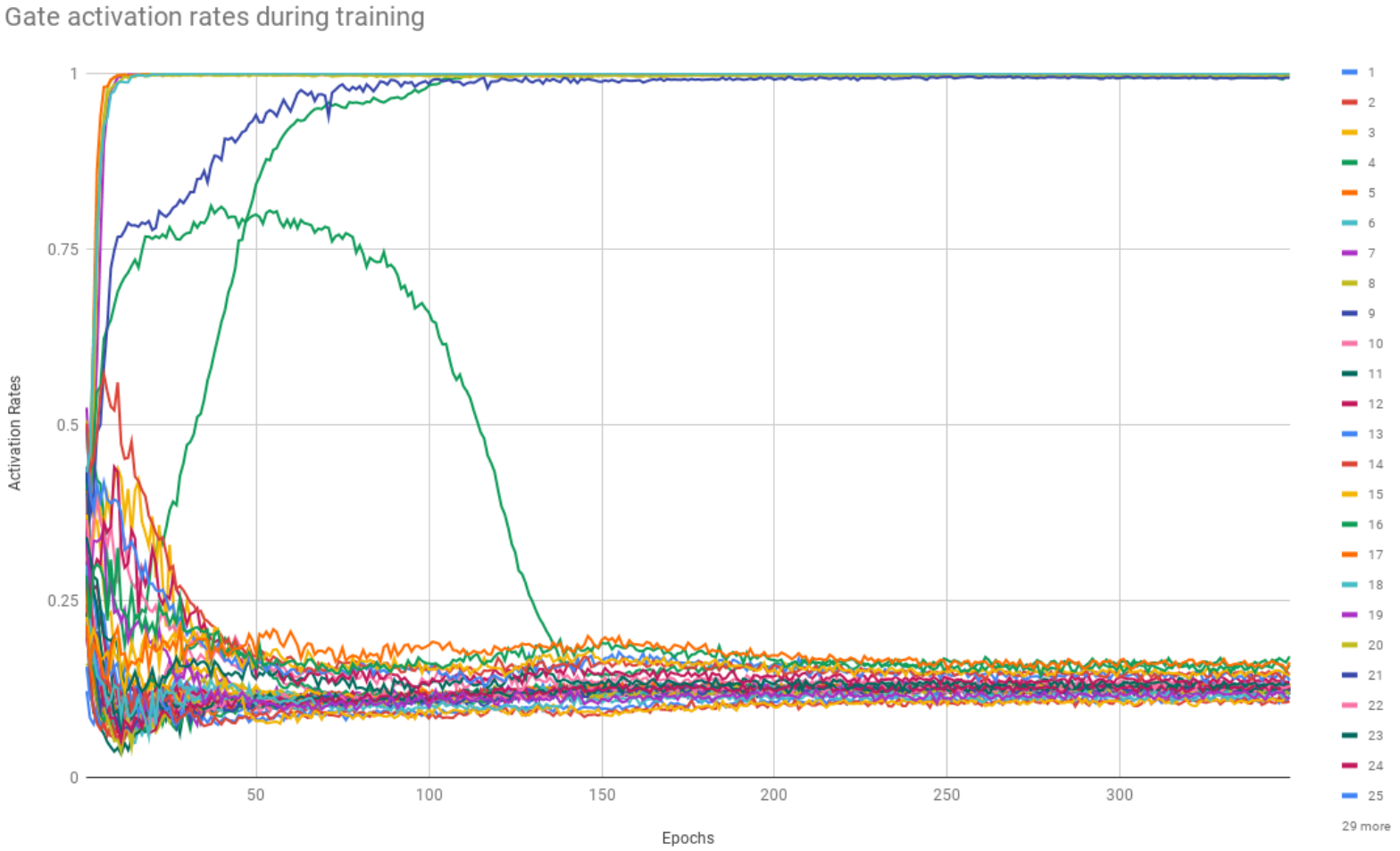

Fig. 12: Layer-based ResNet-110 PB with target rate of 0.2

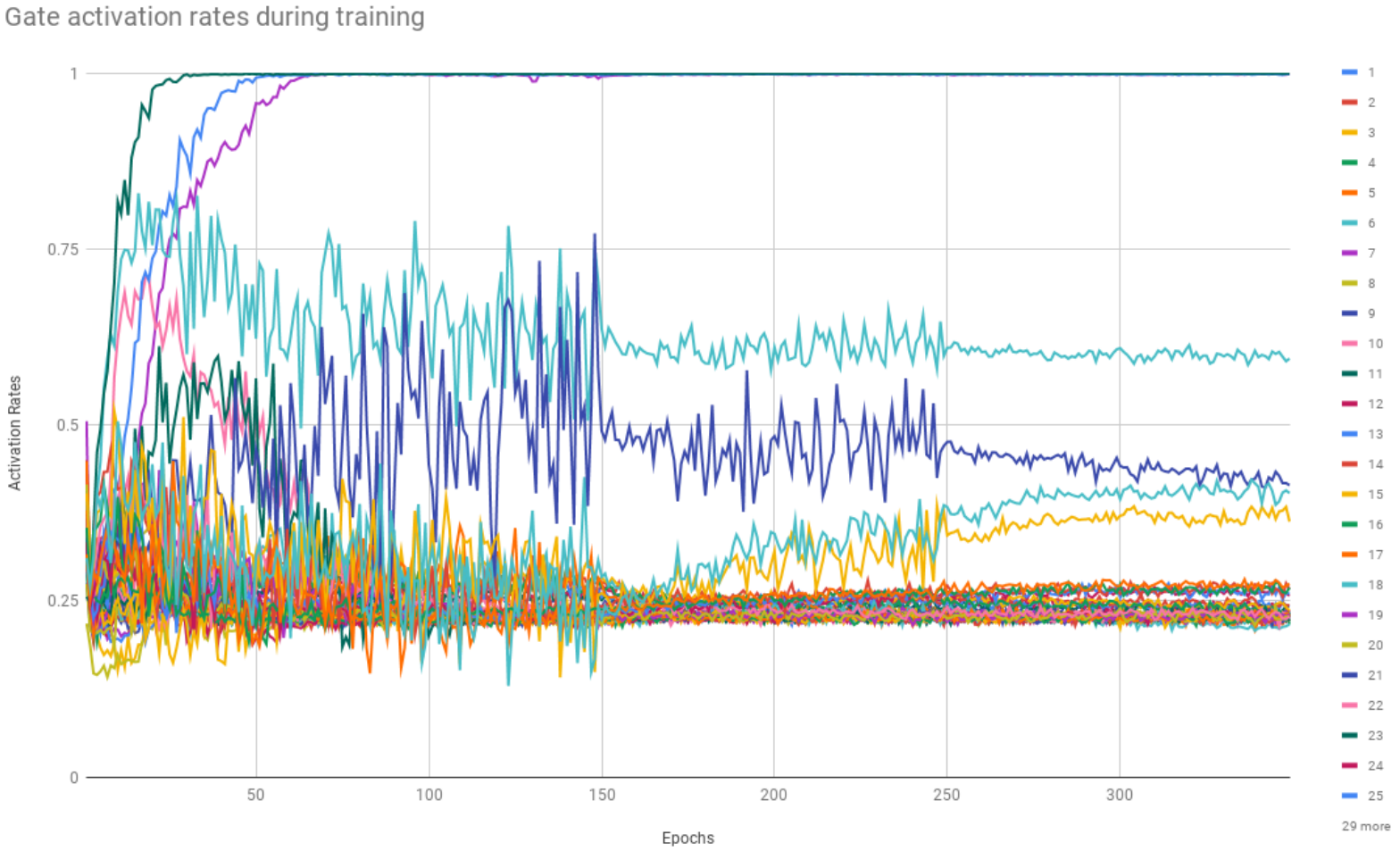

Fig. 13: Layer-based ResNet-110 PG with target rate of 0.2

4.5 Observations regarding gating, classification accuracy, and polarization

Observation 41 *A layer with activation of p can only affect the final accuracy by p.*

We use this observation to suggest that polarization is beneficial for classification loss.

In this paragraph, we provide the high level, conceptual argument and will follow it with a more precise, concrete example. Consider a network with only two layers and the restriction that, on expectation, only one layer should be on. Then let p be the probability that layer 1 is on. Intuitively, if $p \notin \{0, 1\}$, then we are in a high entropy state where the network must deal with a large amount of uncertainty regarding which layers will be active. Furthermore, some percentage of the time no layer will be run at inference time, causing the network to completely fail. Additionally, the percentage of the time that the network tries to train both layers to work together may be "wasted" in some sense, since they will only execute together a small percentage of time during inference.

A more concrete example follows:

Observation 42 *Consider a network with two layers with data-independent probability p_1 and p_2 of being on, restricted to the case that $p_1 + p_2 = 1$. Let a_1 be the expected accuracy of a one-layer network and a_2 be the expected accuracy of a two-layer network. A polarized network ($p_1 \in \{0, 1\}$) will have higher expected accuracy than a not-polarized one if and only if $\frac{a_2}{a_1} \geq 2$.*

Because of the restriction $p_1 + p_2 = 1$, there is only one parameter for the probabilities. Let $p = p_1$. Then $p(1-p)$ is the probability that both layers will be on and also the probability that both layers will be off. Note that the network has a strict upper bound on accuracy of $1 - p(1-p)$ since with probability $p(1-p)$ none of the layers will activate and no output will be given.

Then the expected accuracy of the network for any probability $p \in [0, 1]$ is $(1 - 2p + 2p^2)a_1 + p(1-p)a_2$, note that for $p \in \{0, 1\}$ the accuracy is simply a_1. For a value $p \in (0, 1)$ to have higher expected accuracy, we need

$$\begin{aligned} a_1 &< (1 - 2p + 2p^2)a_1 + p(1-p)a_2 \\ \frac{-2p^2 + 2p}{p(1-p)} &< \frac{a_2}{a_1} \\ 2 &< \frac{a_2}{a_1} \end{aligned}$$

Note that a strong restriction in this case is identical to the coefficient of the batch activation loss being infinite. As the coefficient decreases, the network gains more flexibility and can trade batch activation loss for classification loss, so the argument does not strictly hold in the case described in the paper. However, we believe that the general intuition and statements still apply.

References

1. G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense convolutional networks for efficient prediction. *CoRR, abs/1703.09844*, 2, 2017. 1, 2
2. G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017. 6, 11
3. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2
4. G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661. Springer, 2016. 12
5. S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, pages 2464–2469. IEEE, 2016. 1
6. A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2017. 2, 12