Loss Landscape Sightseeing with Multi-Point Optimization

Ivan Skorokhodov
Neural Networks and Deep Learning Lab
Moscow Institute of Physics and Technology
skorokhodov.is@mipt.ru

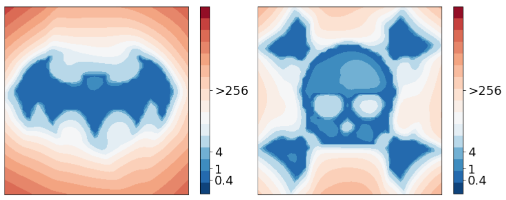
Mikhail Burtsev
Neural Networks and Deep Learning Lab
Moscow Institute of Physics and Technology
burtcev.ms@mipt.ru

Abstract

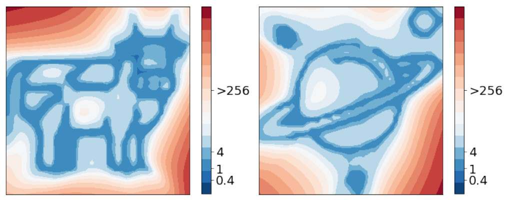
We present *multi-point optimization*: an optimization technique that allows to train several models simultaneously without the need to keep the parameters of each one individually. The proposed method is used for a thorough empirical analysis of the loss landscape of neural networks. By extensive experiments on FashionMNIST and CIFAR10 datasets we demonstrate two things: 1) loss surface is surprisingly diverse and intricate in terms of landscape patterns it contains, and 2) adding batch normalization makes it more smooth. Source code to reproduce all the reported results is available on GitHub[1].

1 Introduction

In this paper, we present *multi-point optimization* (MPO): a technique that allows to find many weight vectors in the parameter space by performing optimization procedure on a considerably smaller amount of parameters, thus saving a lot of memory and computation. This technique allowed us to explore the structure of a loss landscape of neural networks on FashionMNIST and CIFAR10 datasets by finding different landscape patterns (see, for example, Figure 1), and to analyze smoothing capabilities of batch normalization (Ioffe and Szegedy (2015)).



(a) Loss surface on FashionMNIST dataset



(b) Loss surface on CIFAR10 dataset

Figure 1: Examples of a loss landscape of a typical CNN model on FashionMNIST and CIFAR10 datasets found with MPO. Loss values are color-coded according to a logarithmic scale. We used a small VGG-like model which architecture is presented in appendix B; additional visualizations are presented in appendix D. Here we used *test* sets to compute the loss values.

Nguyen (2019) recently showed that, under some mild assumptions, any two minima of a neural network loss surface are connected by a continuous path of approximately the same loss value. Garipov et al. (2018) proposed a practical approach to find such a path. In our work, we extend this approach and instead of finding parameters of a 1D-path between 2 fixed minima we find parameters of a d-dimensional manifold under some generous constraints over K non-fixed weight vectors.

[1] https://github.com/universome/loss-patterns

Part of "Beyond First Order Methods in ML" workshop, NeurIPS 2019, Vancouver, Canada.

2 Related work

Proposed method originates from mode connectivity ideas developed concurrently by Garipov et al. (2018) and Draxler et al. (2018). In those papers, authors empirically demonstrate that any two local minima can be connected by a continuous path of approximately the same loss value. Nguyen (2019) went further and rigorously showed that under very mild assumptions all sublevel sets of loss surface of neural networks are connected. Our approach is based on the optimization procedure used by Garipov et al. (2018) to connect the modes. But instead of finding a single 1D-path between existing 2 weight vectors we fit parameters of a d-dimensional manifold under arbitrary constraint over K weight vectors.

To the best of our knowledge multi-point optimization of neural networks was not previously explored in such a general scenario. The one specific direction in which similar ideas were developed is fast ensembling strategies. For example, Huang et al. (2017); Garipov et al. (2018) recently showed that one can construct a good ensemble by taking models along the SGD trajectory if one carefully perturbs the learning rate during the optimization process to force the exploration of other minima.

Current work does not address the ensembling potential of the proposed method, and we focus on the loss landscape analysis instead. In this sense it is highly related to the work of Li et al. (2018) who performed an empirical investigation of the loss surfaces of large-scale neural networks. It was demonstrated that deep models tend to have a more irregular loss surface and one of the techniques to make it more smooth is to use skip-connections. Some analysis of the loss landscape with mode connectivity ideas was performed by Gotmare et al. (2018) who demonstrated that while plain SGD simply goes down the hill — SGD with warm restarts walks around the barriers. The fact that batch normalization (BN) smoothes the loss landscape is not new and was previously shown by Santurkar et al. (2018). Our work provides one more empirical validation of this fact by using another set of tools.

3 Our method

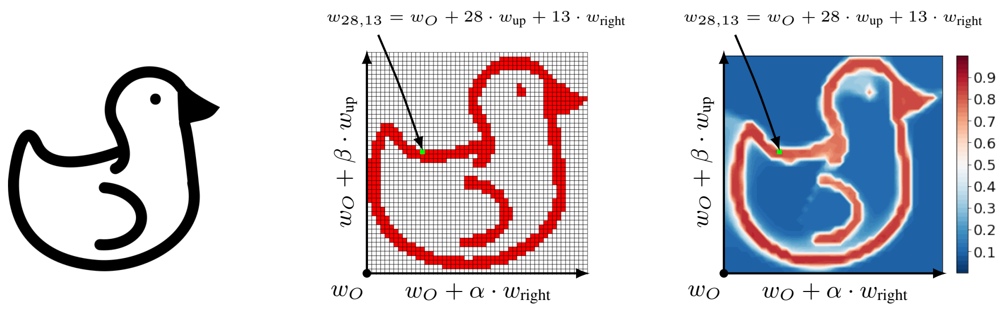
Let the parameter space of the model be $\mathbb{R}^n$. Multi-point optimization works by arranging K weight vectors $\boldsymbol{w}_1, ..., \boldsymbol{w}_K \in \mathbb{R}^n$ on some d-dimensional manifold $M_{\boldsymbol{\theta}} \subset \mathbb{R}^n$ and then optimizing its parameters $\boldsymbol{\theta} \in \Theta$. To avoid unnecessary abstraction and complications, and since in all the presented experiments we set d-dimensional manifold to be just a 2D-plane, hereinafter we describe our method for $M_{\boldsymbol{\theta}} = \mathbb{R}^2 \subset \mathbb{R}^n$. But we emphasize that it is not limited to this setup and can be straightforwardly extended to other structures and dimensionalities.

Optimizing parameters of K weight vectors lying on some 2D-plane is not very exciting and one would like to regularize this optimization in some interesting way. In all our experiments we use a very specific family of constraints over $\boldsymbol{w}_i, i = 1, ..., K$: we force these weight vectors to form some binary 2D-picture. To save space we allow ourselves to describe our method specifically for this constraint family. But we again emphasize that other regularization families can be used. For example, following Rosen (1996), one can force weight vectors to provide models with decorrelated predictions, thus making them form a stronger ensemble.

Now, imagine we want to find weight vectors $\boldsymbol{w}_1, ..., \boldsymbol{w}_K \in M_{\boldsymbol{\theta}}$ such that they form a pattern in the loss landscape depicted on figure 2b. This means that they must lie on the same 2D-plane and be arranged in such a way that their contour map resembles the desired pattern, i.e. some weight vectors correspond to models with low loss values (*black* pixels of the pattern) and others correspond to models with high loss values (*white* pixels). If one looks at a contour map computed around the arbitrary minimum, one generally sees a pit surrounded by random hills and mountains (Li et al. (2018)), but MPO allows to find a region with a nice-looking landscape, like on Figure 2c.

Consider our pattern contains K pixels: this means that we want to find parameters of K weight vectors $\boldsymbol{w}_1, ..., \boldsymbol{w}_K \in \mathbb{R}^n$ with pattern constraints discussed above. Doing so without MPO would be a tedious process since K can be very large (in our experiments we usually have $K = 50^2$). To perform MPO we first parametrize our 2D-plane $M_{\boldsymbol{\theta}}$ with three weight vectors: $(\boldsymbol{w}_O, \boldsymbol{w}_{\text{up}}, \boldsymbol{w}_{\text{right}}) = \boldsymbol{\theta} \in \mathbb{R}^{3n}$. Next, for each $\boldsymbol{w}_i$ we associate a pair of numbers $(\alpha, \beta) \in \mathbb{N}^2$, which correspond to (α, β)-th pixel of the picture 2a, and which specifies its position on $M_{\boldsymbol{\theta}}$:

$$\boldsymbol{w}_i = \boldsymbol{w}_{\alpha,\beta} = \boldsymbol{w}_O + \alpha \cdot \boldsymbol{w}_{\text{right}} + \beta \cdot \boldsymbol{w}_{\text{up}}. \quad (1)$$



(a) Original pattern we want to find in the weight space

(b) Parametrization of the pattern downsampled to 50×50 size

(c) Accuracy landscape of the found minimum with our method

Figure 2: Multi-point optimization method for 2D pattern fitting on FashionMNIST dataset.

Since parametrization (1) is just a linear combination of some vectors we can compute the gradients of any differentiable loss function $\mathcal{L}(\boldsymbol{w}_{\alpha,\beta})$ with respect to $\boldsymbol{\theta}$. And since for each model we have to keep only its 2D-coordinates instead of the corresponding large n-dimensional weight vector we obtain substantial memory savings.

We would like the coordinate system, defined by $\boldsymbol{w}_O, \boldsymbol{w}_{\text{up}}, \boldsymbol{w}_{\text{right}}$, to be orthogonal and properly scaled. For this we enforce $\boldsymbol{w}_{\text{up}} \perp \boldsymbol{w}_{\text{right}}$ and $\|\boldsymbol{w}_{\text{up}}\| = \|\boldsymbol{w}_{\text{right}}\|$. We introduce a new parameter vector $\boldsymbol{\phi}_{\text{right}} \in \mathbb{R}^n$ instead of $\boldsymbol{w}_{\text{right}}$ and optimize for it. Vector $\boldsymbol{w}_{\text{right}}$ is directly computed from $\boldsymbol{\phi}_{\text{right}}$ and $\boldsymbol{w}_{\text{up}}$ via Gram-Shmidt orthogonalization process:

$$\hat{\boldsymbol{w}}_{\text{right}} = \boldsymbol{\phi}_{\text{right}} - \frac{\langle \boldsymbol{\phi}_{\text{right}}, \boldsymbol{w}_{\text{up}} \rangle}{\|\boldsymbol{w}_{\text{up}}\|^2} \cdot \boldsymbol{w}_{\text{up}}, \qquad \boldsymbol{w}_{\text{right}} = \frac{\|\hat{\boldsymbol{w}}_{\text{up}}\|}{\|\hat{\boldsymbol{w}}_{\text{right}}\|} \hat{\boldsymbol{w}}_{\text{right}}. \tag{2}$$

One can easily note that it is differentiable with respect to $\boldsymbol{\phi}_{\text{right}}$, that's why we have no trouble to optimize it via gradient methods. To fit a pattern we minimize the cross-entropy in black points and maximize it in white points, i.e. we optimize the following functional:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^{T} \left[\frac{1}{|P_+|} \sum_{(\alpha,\beta) \in P_+} \log p_{\boldsymbol{w}_{\alpha,\beta}}(\boldsymbol{y}^{(t)} | \boldsymbol{x}^{(t)}) - \frac{1}{|P_-|} \sum_{(\alpha,\beta) \in P_-} \log p_{\boldsymbol{w}_{\alpha,\beta}}(\boldsymbol{y}^{(t)} | \boldsymbol{x}^{(t)}) \right], \tag{3}$$

where P_-, P_+ are sets of indices of black and white pixels of a pattern respectively, $\{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ are training pairs and $p_{\boldsymbol{w}_{\alpha,\beta}}(\boldsymbol{y}|\boldsymbol{x})$ is our model at a point $\boldsymbol{w}_{\alpha,\beta}$. We optimize for $\boldsymbol{\theta}$ with standard gradient methods and the details of this procedure are specified in appendix A.

4 Experiments

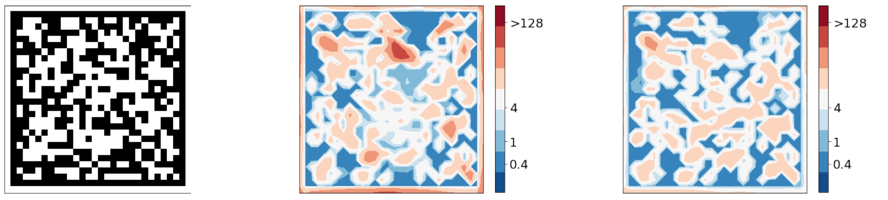
4.1 Finding patterns of the loss landscape

Intuitively, a loss surface with irregular and complicated landscape should be hard to optimize. In the first series of experiments, we explore how diverse and sophisticated it can be by searching for different patterns in the loss landscape of neural networks. We take a simple VGG-like CNN architecture (described in appendix B) and optimize the parameters of a 2D-plane described in section 3 on FashionMNIST and CIFAR10 datasets to find a specific pattern.

To our surprise, literally, any pattern of an adequate size that we tried was found with high quality in the loss landscape. Heat maps of the found 2D-planes are presented in Figure 1 and in appendix D. These results demonstrate that the loss surface of neural networks can be very curved and ill-behaved. Interestingly, for FashionMNIST dataset resulting patterns evaluated on a train set are almost indistinguishable from the patterns evaluated on a test set. This indicates that loss surface diversity is a property of the whole risk functional, not only its empirical counterpart.

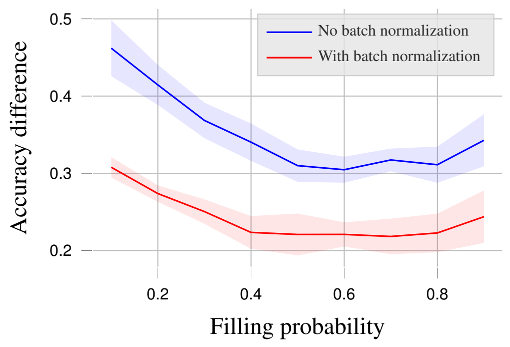
4.2 Smoothing properties of batch normalization

In the second batch of experiments, we study the smoothing properties of batch normalization. For this, we generate random binary masks of size 30×30. We vary the probability of pixels being

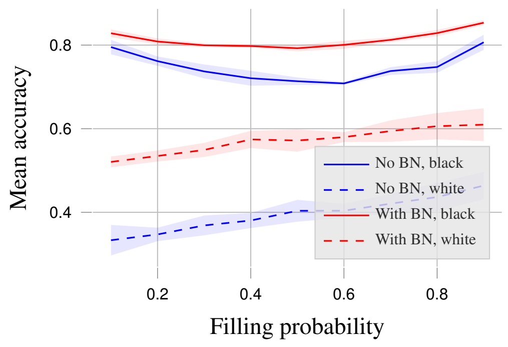


(a) Mask of the pattern (b) Without batch normalization (c) With batch normalization

Figure 3: (a) Example of a random binary mask with a filling probability of 0.5. (b) The result of the MPO procedure for the model without batch normalization. (c) The result of the MPO procedure for the model with batch normalization.



(a) The difference in accuracy. Standard deviations are taken across 5 runs.



(b) Mean accuracy for different values of filling probability. Blue — without and red — with batch normalization. A solid line — for black pixels and a dotted one — for white pixels.

masked from 0.1 to 0.9, thus obtaining the patterns of varying complexity. For the probability of 0.1 and 0.9, the landscape is mostly homogeneous, being either completely black or completely white. For probabilities near 0.5, we get very irregular patterns that are very difficult to fit (see Figure 3a).

For each random pattern, we've trained two types of CNN models on FashionMNIST dataset, one with batch normalization and another without it. We then measured the mean accuracy in black and white pixels on a test set and took the difference. The difference in accuracy across 5 runs with the corresponding standard deviation is depicted in Figure 4a. As one can see, it's much more difficult for a model with batch normalization to fit the desired pattern. This indicates that such complex patterns as random binary masks are less likely to occur on its loss surface, implying that it is more smooth and regular. And as Figure 4b shows models trained with batch normalization generally attain higher scores — a consequence of a more decent loss landscape.

We note that batch normalization is not well aligned with our parametrization of 2D-plane, so we used a small trick to alleviate this. The problem and the solution are discussed in appendix C.

pattern说明loss surface的irregular，加了BN过后，surface更加规整了，所以pattern更难去fit

5 Conclusion

We presented a method that allows one to fit several points in the parameter space by running an optimization procedure on a significantly smaller amount of parameters. It was used to demonstrate that the loss surface of neural networks conceals regions with arbitrary landscape patterns, implicating its diversity and optimization complexity. Besides, along the way, we showed that batch normalization makes it more regular. It is important to note, however, that though our approach provides drastic memory savings, it does not give notable benefits in terms of clock-wall time since on each iteration we perform many forward passes to update the parameters of the underlying manifold. In current submission, we didn't explore its use to construct better, more decorrelated ensembles, but we believe it to be a very fruitful research direction with potential practical applications. Theoretical analysis was also left for future work.

Acknowledgments

The work was supported by National Technology Initiative and PAO Sberbank project ID 0000000007417F630002. All the icons used in our work were taken from www.flaticon.com. We thank Eugene Golikov and Mikhail Arkhipov for valuable discussions and suggestions.

References

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/draxler18a.html`.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8789–8798. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/8095-loss-surfaces-mode-connectivity-and-fast-ensembling-of-dnns.pdf`.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Using mode connectivity for loss landscape analysis. *CoRR*, abs/1806.06977, 2018. URL `http://arxiv.org/abs/1806.06977`.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL `https://openreview.net/forum?id=BJYwwY9ll`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. URL `http://dl.acm.org/citation.cfm?id=3045118.3045167`.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf`.

Quynh Nguyen. On connected sublevel sets in deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4790–4799, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/nguyen19a.html`.

Bruce E Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996. doi: 10.1080/095400996116820. URL `https://doi.org/10.1080/095400996116820`.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2483–2493. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf`.

A Optimization details

First, we downsample an image to a size $w \times h$ (usually 50×50). Then we initialize the parameters of each layer for weights $\boldsymbol{w}_O, \boldsymbol{w}_{\text{up}}$ and ϕ_{right} via Xavier initialization. We also found it beneficial to scale vectors $\boldsymbol{w}_{\text{up}}, \boldsymbol{w}_{\text{right}}$ by some small scale value s, initializing it somewhere between 0.01 and 0.1 depending on the size of a pattern. We found it important to optimize the scaling factor $s \in \mathbb{R}$ as well, since adjusting it manually is a difficult process which requires extensive hyperparameters tuning. So finally we minimize (3) for $\boldsymbol{\theta} = \{\boldsymbol{w}_O, \boldsymbol{w}_{\text{up}}, \boldsymbol{\phi}_{\text{right}}, s\}$ via Adam optimizer. On each iteration, we pick random subsets of P_- and P_+ to compute the gradients to make the training procedure faster.

Also during training, we do not use the pixels which are surrounded by the pixels of the same class to reduce computation since we found that model interpolates between neighboring values without being trained to do so.

Batch Normalization will not work for this procedure as it is, because during orthogonalization process scale parameters become negative. To make it valid we add -0.5 for its scale parameter to make it have zero mean and add 0.5 back during the forward pass. Details about this trick are provided in the appendix C.

B Experiment details

For experiments for finding pictures in the loss landscape, our architecture consisted of 3 convolutional layers with kernel sizes of 3 and 8, 32, 64 number of channels and ReLU non-linearity after each layer. Then we used adaptive average pooling to convert the representation to $64 \times 4 \times 4$ tensor, flattened it and passed to a one-layer MLP with 128 hidden units.

For experiments with batch normalization we used a very similar architecture, the only difference was that we had more convolutional layers with 8, 8, 32, 32, 64, 64 number of channels. We had batch normalization after each convolutional layer (i.e. before activation is applied) for a model with batch normalization.

We used Adam optimizer with a learning rate of 0.0003. We thresholded cross-entropy value in white cells by a value of 2.5: otherwise it was too easy for the model just to have a very large cross-entropy value everywhere to get a good optimization loss value. We trained the model with a batch size of 512 and used 50 cells per update in each iteration. We initialize scale value s at 0.1.

C Batch Normalization reparametrization

Consider we have two random vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ and we want to perform a Gram-Shmidt orthogonalization to get vector $\boldsymbol{w}$ such that $\boldsymbol{w} \perp \boldsymbol{u}$. If $\boldsymbol{u}, \boldsymbol{v}$ come from zero-mean normal distribution with a small variance, their elements are i.i.d and n is large enough, then $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ will have zero mean with a small variance. Such an initialization scheme is quite typical for most of the layers, but not for normalization ones, like Batch Normalization. This is because scale parameters $\boldsymbol{s}$ of BN layer are initialized from $U[0, 1]$ (or even all ones). As a result, the scalar product of the parameters of two BN layers is much larger than zero, which, in turn, results in vector $\boldsymbol{w}$ having a lot of negative values:

$$\boldsymbol{w} = \boldsymbol{v} - \frac{\langle \boldsymbol{v}, \boldsymbol{u} \rangle}{\|\boldsymbol{u}\|^2} \boldsymbol{u} \tag{4}$$

Such a behavior is not fatal, but not desirable, because scale parameter associates with standard deviation, so this is an abnormal situation when it's negative.

To alleviate this issue we parametrize batch normalization in such a way that scale parameter has zero mean. We do this simply by initializing it from $U[-0.5, 0.5]$ instead of $U[0, 1]$. Since this makes it contain negative values, during the forward pass we have to add 0.5 back to make it all-positive again.

D Additional visualizations of loss and accuracy surfaces

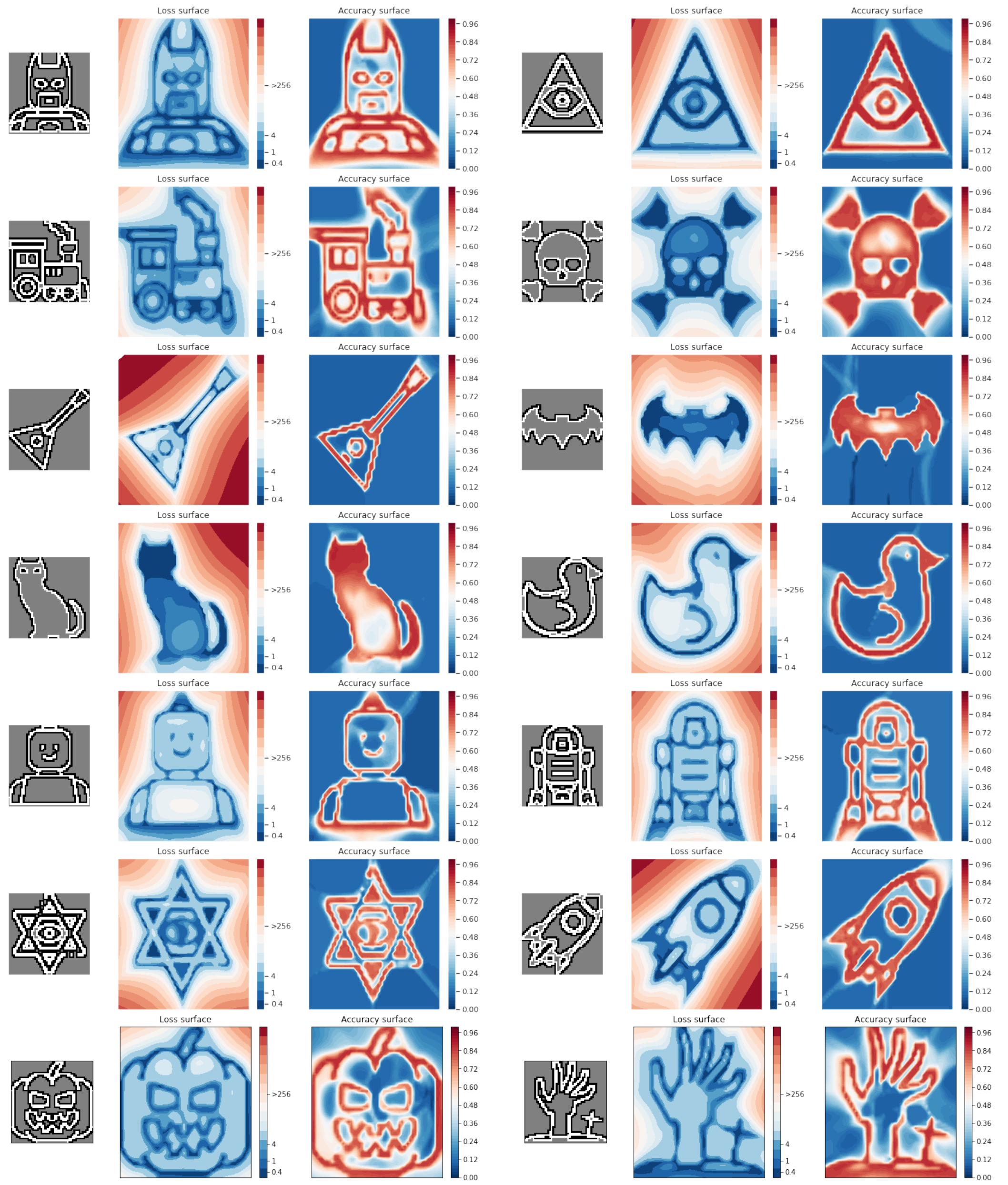


Figure 5: Additional results for pattern search on FashionMNIST dataset. Since train and test landscapes are almost visually indistinguishable for our model in the case of FashionMNIST dataset, we depict here only *test* loss surfaces.

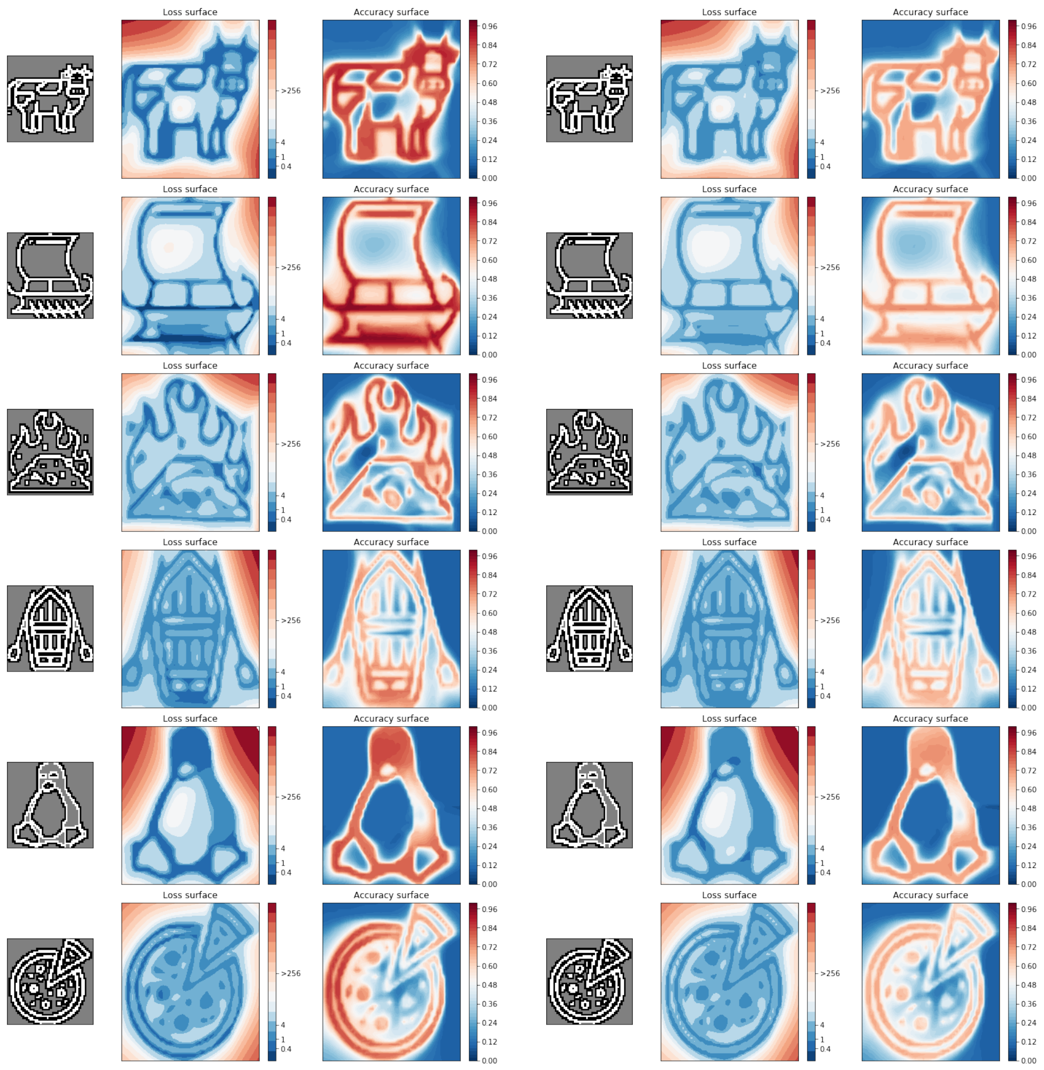


Figure 6: Additional results for pattern search on CIFAR10 dataset. Left column depicts train loss/accuracy surface, right column depicts test loss/accuracy surface.