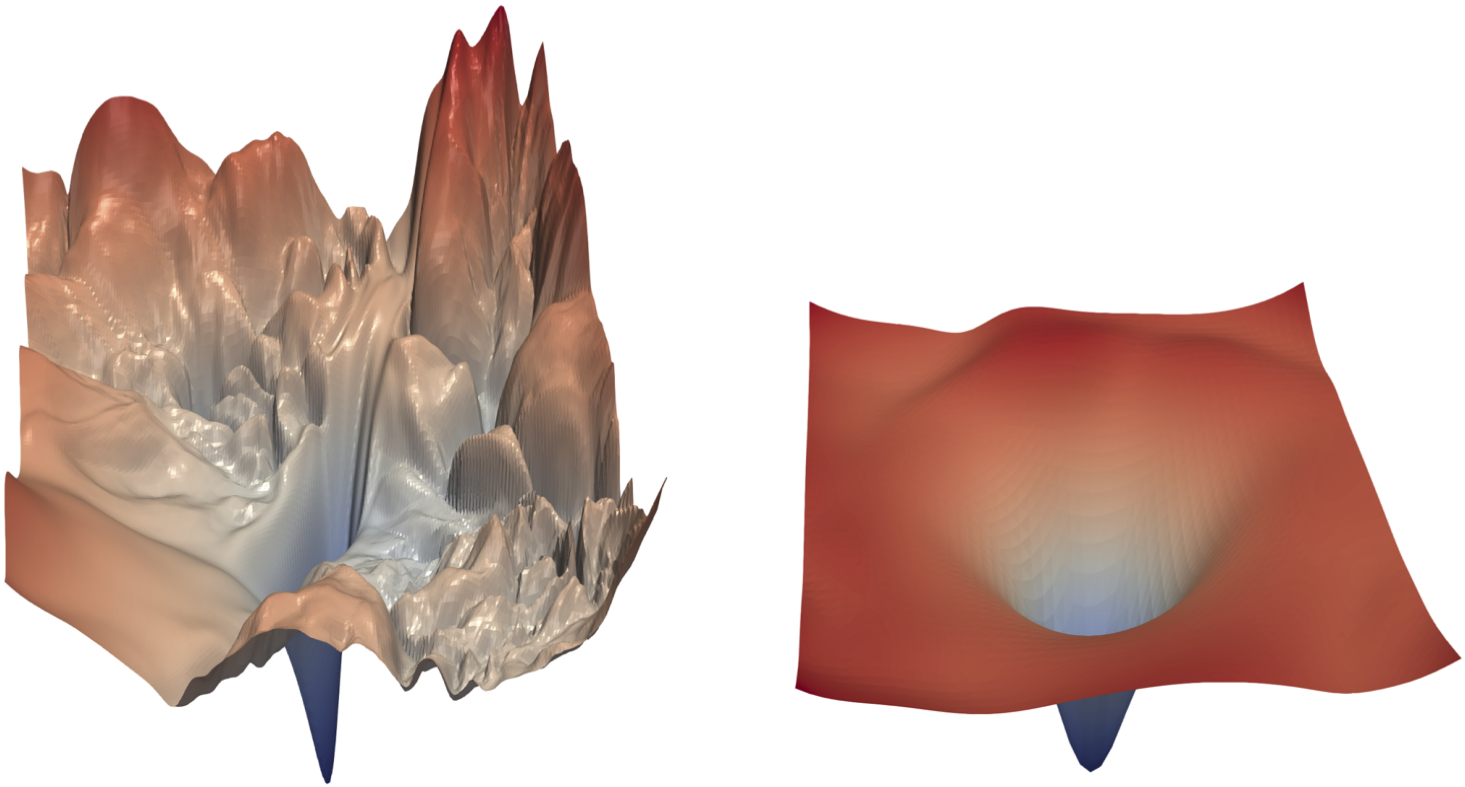
Outline for today

- Visualisation of some network loss landscapes
  - ResNet-56 with and without skip connection
  - VGG9 dependence on weight decrease and normalisation
  - ResNet depth dependence and skip connections
- Early theoretical results on over parameterisation and number of connected components of loss level curves.
- Removing the weight nonlinearity through convexification.

Loss landscape example: ResNet-56 (Li et al. 18'[1])



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

[1] http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Loss landscape example: VGG9 (Li et al. 18'[2])

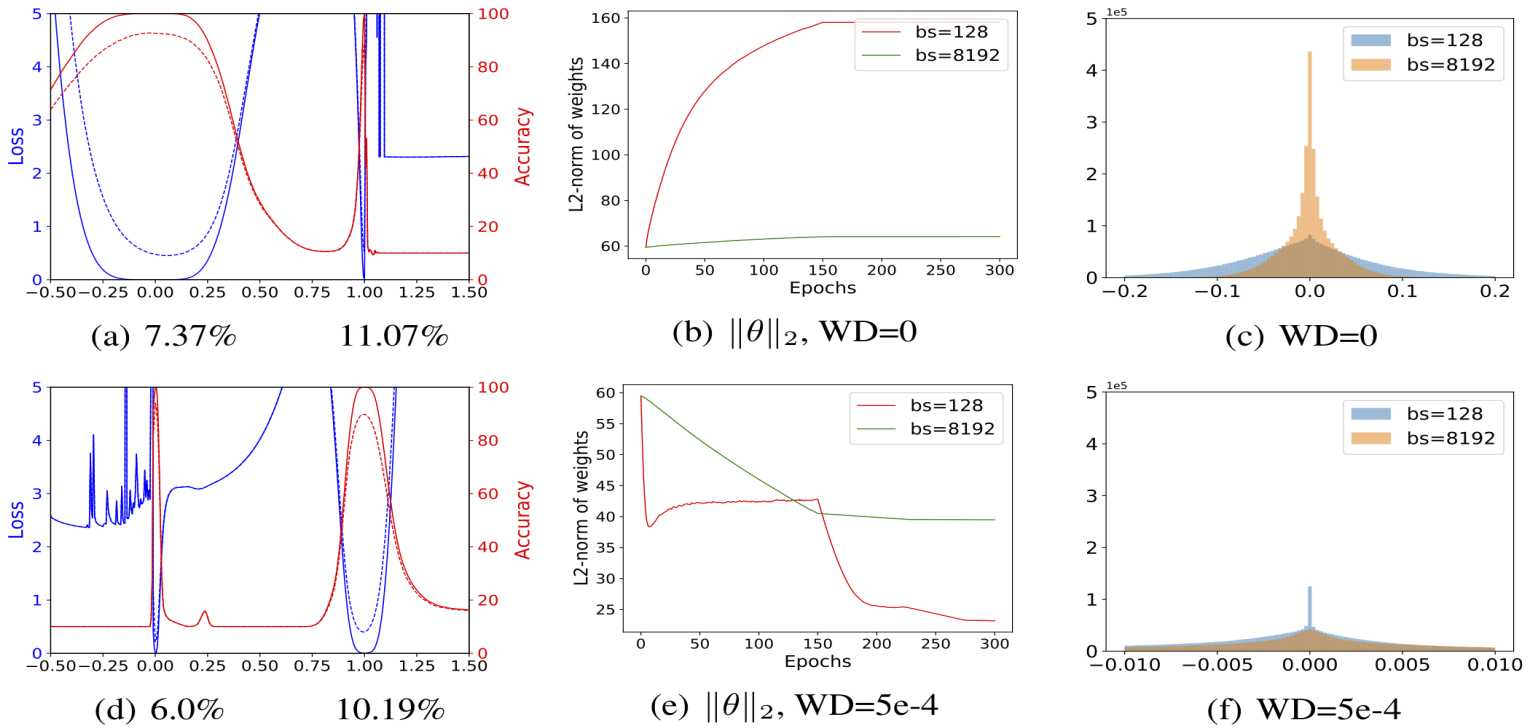


Figure 2: (a) and (d) are the 1D linear interpolation of VGG-9 solutions obtained by small-batch and large-batch training methods. The blue lines are loss values and the red lines are accuracies. The solid lines are training curves and the dashed lines are for testing. Small batch is at abscissa 0, and large batch is at abscissa 1. The corresponding test errors are shown below. (b) and (e) shows the change of weights norm $||\theta||_2$ during training. When weight decay is disabled, the weight norm grows steadily during training without constraints (c) and (f) are the weight histograms, which verify that small-batch methods produce more large weights with zero weight decay and more small weights with non-zero weight decay.

[2] http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Loss landscape example: VGG9 normalized (Li et al. 18'[3])

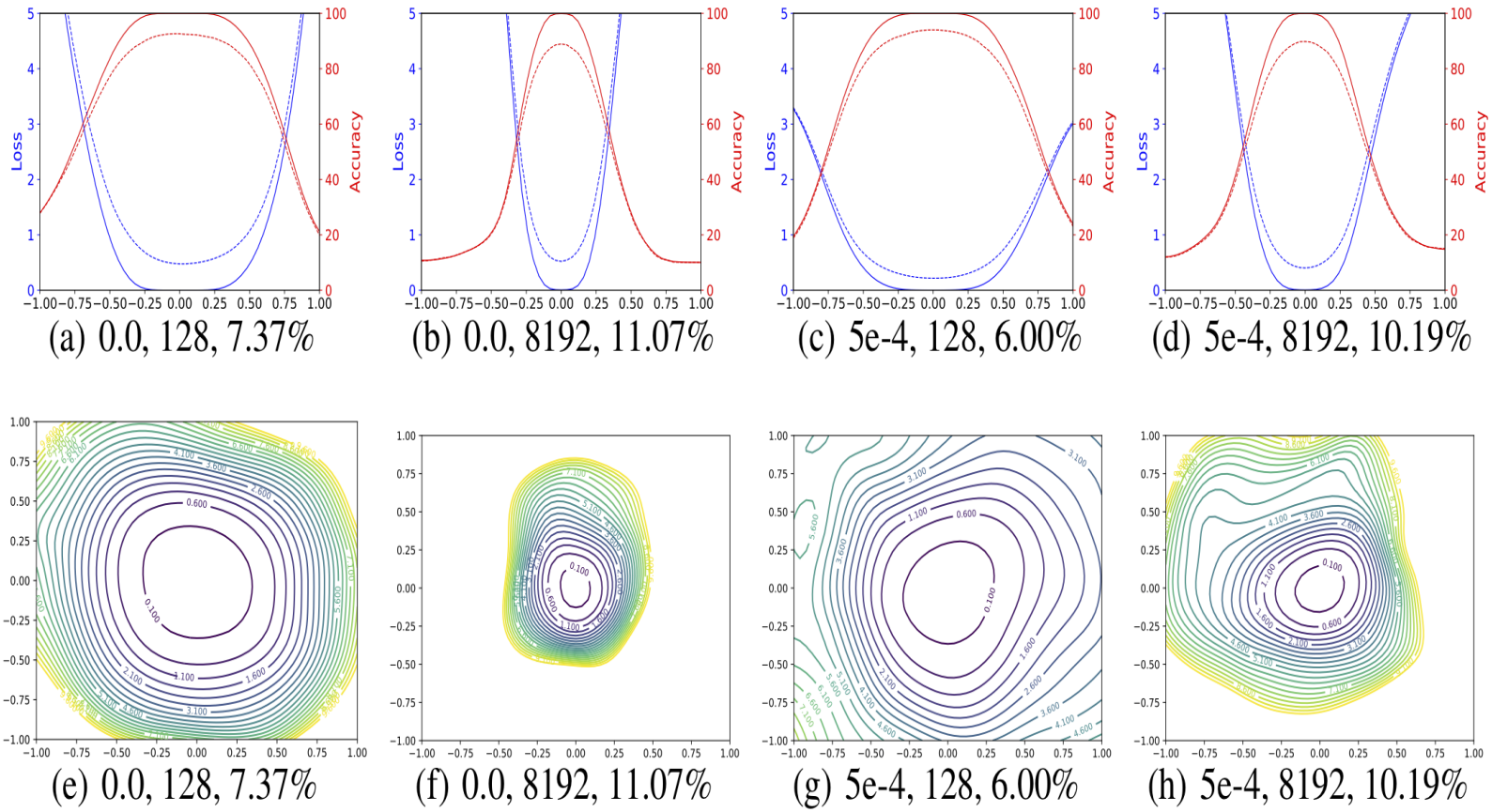
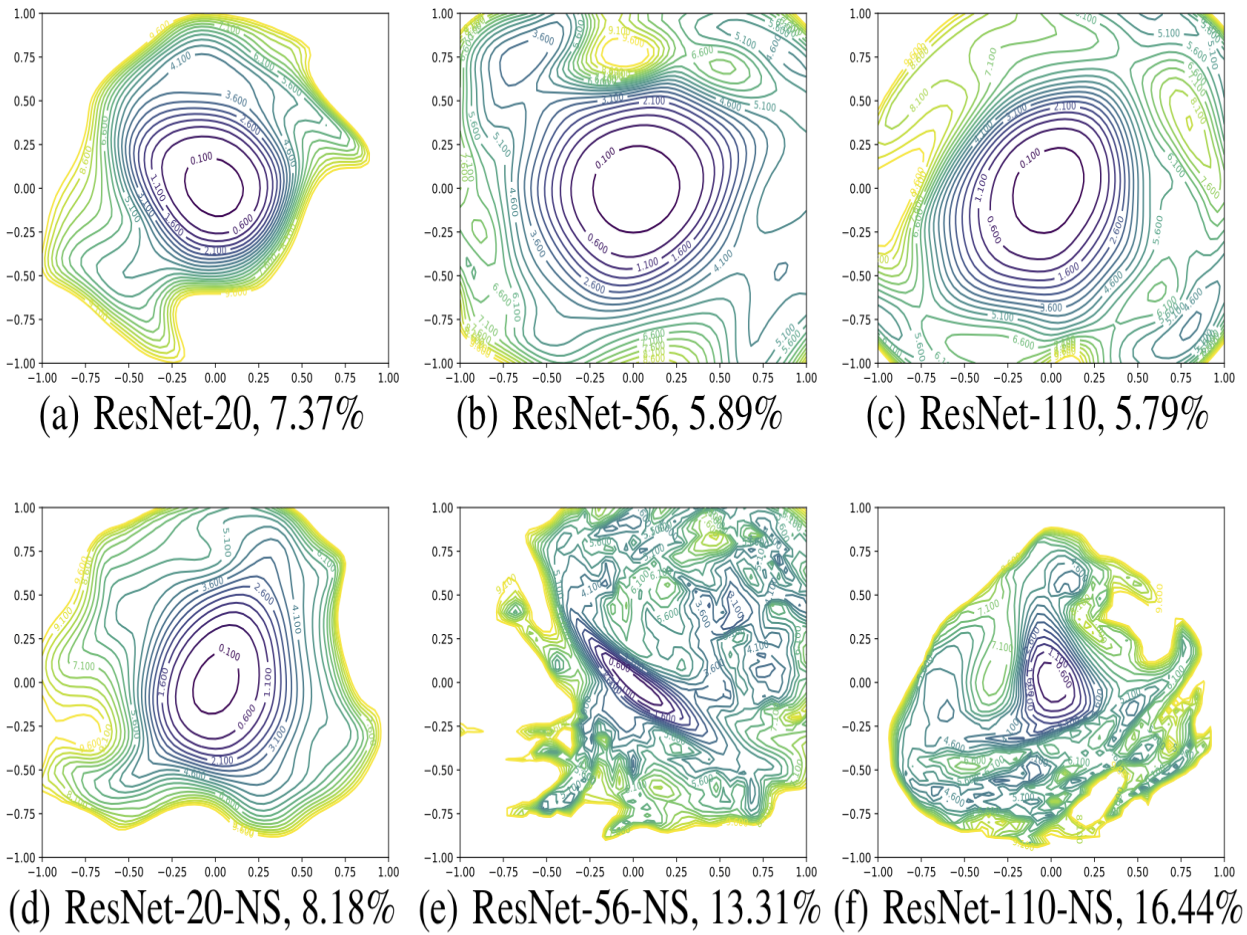


Figure 3: The 1D and 2D visualization of solutions obtained using SGD with different weight decay and batch size. The title of each subfigure contains the weight decay, batch size, and test error.

[3] http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Loss landscape example: ResNet skip (Li et al. 18'[4])



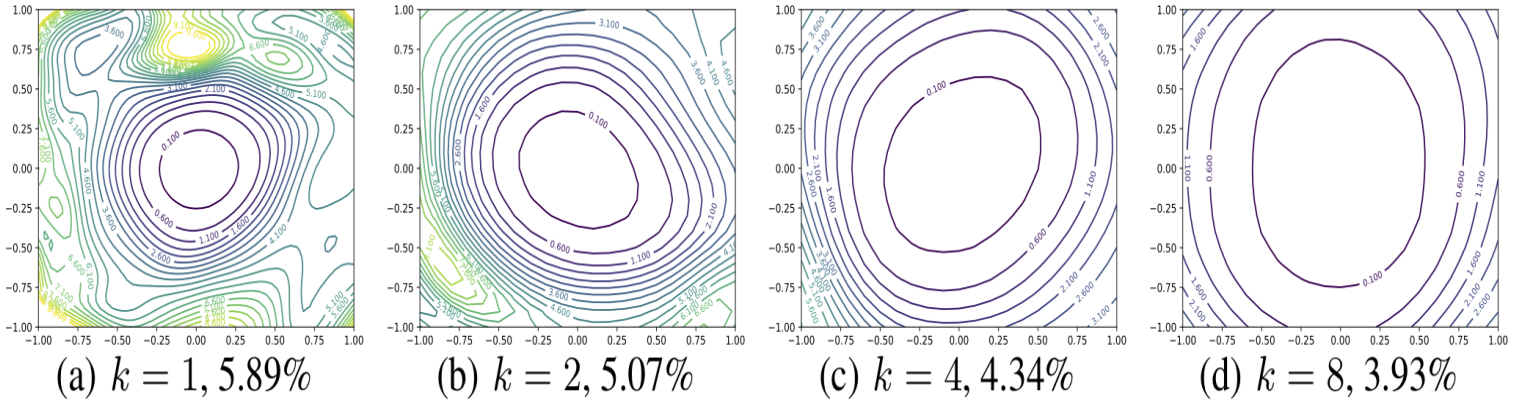
(a) ResNet-20, 7.37% (b) ResNet-56, 5.89% (c) ResNet-110, 5.79%

(d) ResNet-20-NS, 8.18% (e) ResNet-56-NS, 13.31% (f) ResNet-110-NS, 16.44%

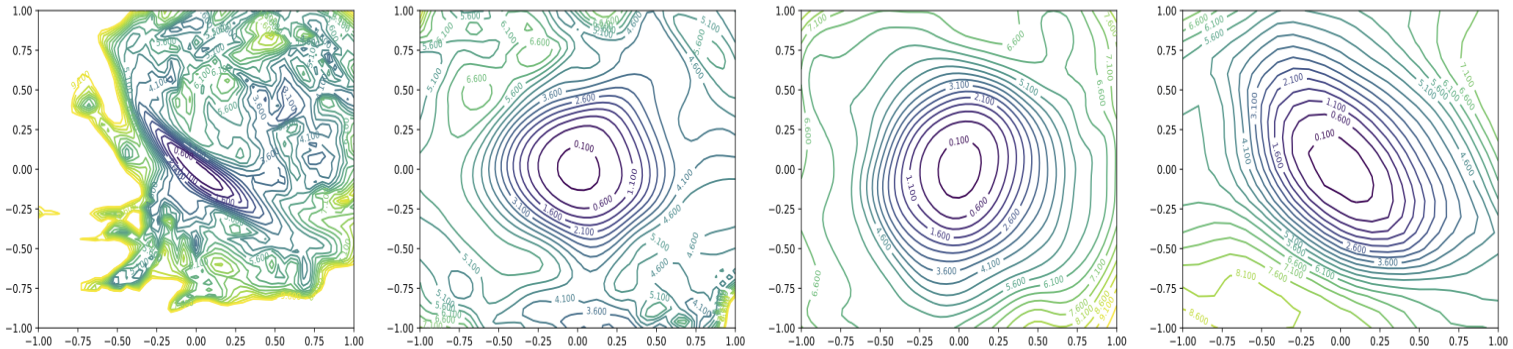
Figure 5: 2D visualization of the loss surface of ResNet and ResNet-noshort with different depth.

[4] http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Loss landscape example: ResNet width (Li et al. 18'[5])



(a) $k = 1$, 5.89% (b) $k = 2$, 5.07% (c) $k = 4$, 4.34% (d) $k = 8$, 3.93%



(e) $k = 1$, 13.31% (f) $k = 2$, 10.26% (g) $k = 4$, 9.69% (h) $k = 8$, 8.70%

Figure 6: Wide-ResNet-56 on CIFAR-10 both with shortcut connections (top) and without (bottom). The label $k = 2$ means twice as many filters per layer. Test error is reported below each figure.

[5] http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Topology of loss landscape (Freeman et al. 16'[6])

Consider our loss function: $\mathcal{L}(\theta; X, Y) = n^{-1} \sum_{\mu=1}^{n} l(\theta; x_\mu, y_\mu)$ and its associated level set

$$\Omega_{\mathcal{L}}(\lambda) = \{\theta : \ \mathcal{L}(\theta; X, Y) \leq \lambda\}$$

Of particular interest are the number of connected components, say N_λ, in $\Omega_{\mathcal{L}}(\lambda)$. If $N_\lambda = 1$ for all λ then $\mathcal{L}(\theta; X, Y)$ has no isolated local minima and any descent method can obtain a global minima.

If $N_\lambda > 1$ there may be "spurious valleys" in which the minima in the connected component does not achieve the global minima.

[6] https://arxiv.org/pdf/1611.01540.pdf

Topology of loss landscape (Freeman et al. 16'[7])

Linear network: single component

Let $H(x;\theta)$ be an L layer net given by $h^{(\ell)} = W^{(\ell)}h^{(\ell-1)})$ with $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, then if $n_\ell > \min(n_0, n_L)$ for $0 < \ell < L$, the sum of squares loss function has a single connected component

ReLU network: multiple components

Let $H(x;\theta)$ be an L layer net given by $h^{(\ell)} = \sigma(W^{(\ell)}h^{(\ell-1)}))$ with $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $\sigma(\cdot) = \max(0, \cdot)$, then for any choice of n_ℓ there is a distribution of data (X, Y) such that there are more than one single connected component.

[7] https://arxiv.org/pdf/1611.01540.pdf

Topology of loss landscape: quadratic activation (Venturi et al. 16'[8])

ReLu activation network: nearly connected

Consider a 2 layer ReLu network $H(x,\theta) = W^{(2)}\sigma(W^{(1)}x)$ with $W^{(1)} \in \mathbb{R}^{m\times n}$ and $W^{(2)} \in \mathbb{R}^m$, then for any two parameters θ_1 and θ_2 with $\mathcal{L}(\theta_i) \leq \lambda$ for $i = 1, 2$, then there is a path $\gamma(t)$ between θ_1 and θ_2 such that $\mathcal{L}(\theta_{\gamma(t)}) \leq \max(\lambda, m^{-1/n})$.

quadratic activation network: single component

Let $H(x,\theta)$ be an L layer net given by $h^{(\ell)} = \sigma(W^{(\ell)}h^{(\ell-1)})$ with $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and quadratic activation $\sigma(z) = z^2$, then once the number of parameters $n_\ell \geq 3N^{2^\ell}$ where N is the number of data entries, then the sum of squares loss function has a single connected component. For the two layer case with a single quadratic activation this simplifies to $n > 2N$.

[8] https://arxiv.org/pdf/1802.06384.pdf

Convexifying the parameters pt. 1 (Zhang et al. 16'[9])

Consider a two layer convolutional neural network composed of one convolutional layer followed by a fully connected layer.
Rather than working with x directly, form P vectors $z_p(x)$ for $p = 1, \ldots, P$ where $z_p(x)$ is the portion of x on patch p of the convolutional layer. Then the k^{th} component of $H(x, \theta)$ is given by

$$H(x,\theta)_k = \sum_{j=1}^{r} \sum_{p=1}^{p} \alpha_{k,j,p} \sigma(w_j^T z_p(x)).$$

Alternatively if we exclude the nonlinearity we can express this sum by

$$\sum_{j=1}^{r} \sum_{p=1}^{p} \alpha_{k,j,p} \sigma(w_j^T z_p(x)) = \sum_{j=1}^{r} Z(x) w_j$$

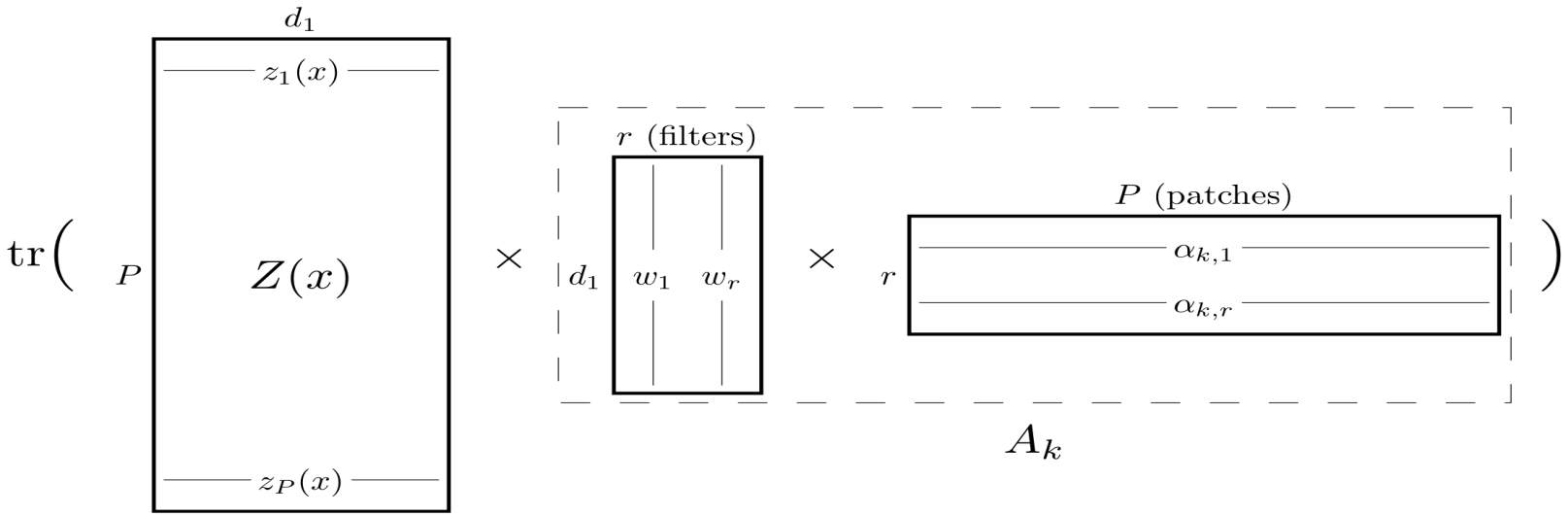
where $Z(x)$ has $z_p(x)$ as its p^{th} row.

[9] https://arxiv.org/pdf/1609.01000.pdf

Convexifying the parameters pt. 2 (Zhang et al. 16'[10])

Using the trace formula this can be further condensed to

$$H(x,\theta)_k = \mathrm{tr}\left(Z(x)\left(\sum_{j=1}^{r} w_j \alpha_{k,j}^T\right)\right) = \mathrm{tr}\left(Z(x)A_k\right)$$



The network parameters are given by A_k nonlinearity is imposed by the A_k having rank r, and we can express all of the parameters of the matrix by A which is similarly rank r.

[10] https://arxiv.org/pdf/1609.01000.pdf

Convexifying the parameters pt. 3 (Zhang et al. 16'[11])

One can impose the network structure through A, but remove the non-convex rank constraint by replacing a convexification, that is the sum of the singular values of A (Schatten-1, or nuclear, norm).

If the convolutional filters and fully connected rows are uniformly bounded in ℓ^2 by B_1 and B_2 respectively, then one can replace then the sum of the singular values of A are bounded by $B_1 B_2 r\sqrt{n}$ where n is the network output dimension and the network parameters can be considered by varying the nuclear norm bound between 0 and $B_1 B_2 r\sqrt{n}$.

The resulting learning programme is fully convex and can be efficiently solved. The above can be extended to nonlinear activations and multiple layers, learning one layer at a time.

[11] `https://arxiv.org/pdf/1609.01000.pdf`

Convexified CNN: MNIST (Zhang et al. 16'[12])

	basic	rand	rot	img	img+rot
SVM_{rbf} [44]	3.03%	14.58%	**11.11%**	22.61%	55.18%
NN-1 [44]	4.69%	20.04%	18.11%	27.41%	62.16%
CNN-1 (ReLU)	3.37%	9.83%	18.84%	14.23%	45.96%
CCNN-1	**2.38%**	**7.45%**	13.39%	**10.40%**	**42.28%**
TIRBM [38]	-	-	**4.20%**	-	35.50%
SDAE-3 [44]	2.84%	10.30%	9.53%	16.68%	43.76%
ScatNet-2 [8]	1.27%	12.30%	7.48%	18.40%	50.48%
PCANet-2 [9]	**1.06%**	6.19%	7.37%	10.95%	35.48%
CNN-2 (ReLU)	2.11%	5.64%	8.27%	10.17%	32.43%
CNN-2 (Quad)	1.75%	5.30%	8.83%	11.60%	36.90%
CCNN-2	1.38%	**4.32%**	6.98%	**7.46%**	**30.23%**

Table 1: Classification error on the basic MNIST and its four variations. The best performance within each block is bolded. The tag "ReLU" and "Quad" means ReLU activation and quadratic activation, respectively.

[12] `https://arxiv.org/pdf/1609.01000.pdf`

Convexified CNN: CIFAR10 (Zhang et al. 16'[13])

	Error rate
CNN-1	34.14%
CCNN-1	**23.62%**
CNN-2	24.98%
CCNN-2	**20.52%**
$\text{SVM}_{\text{Fastfood}}$ [27]	36.90%
PCANet-2 [9]	22.86%
CKN [30]	21.70%
CNN-3	21.48%
CCNN-3	**19.56%**

Table 3: Classification error on the CIFAR-10 dataset. The best performance within each block is bolded.

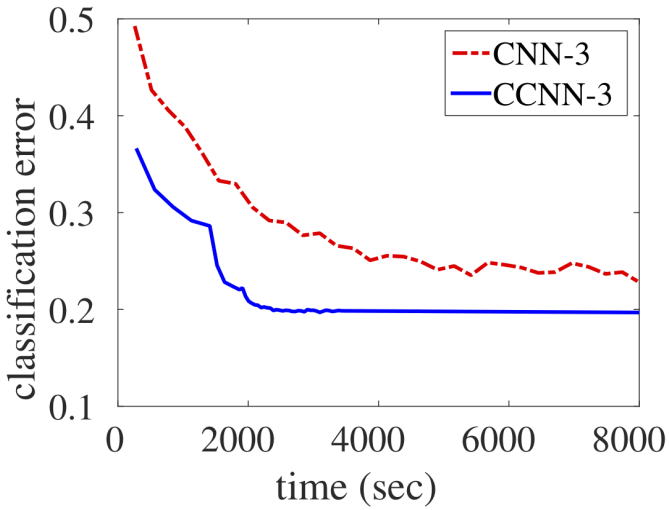


Figure 4: The convergence of CNN-3 and CCNN-3 on the CIFAR-10 dataset.

	CNN-1	CNN-2	CNN-3
Original	34.14%	24.98%	21.48%
Convexified	**23.62%**	**21.88%**	**18.18%**

Table 4: Comparing the original CNN and the one whose top convolution layer is convexified by CCNN. The classification errors are reported on CIFAR-10.

[13] https://arxiv.org/pdf/1609.01000.pdf