Deep learning

9.2. Looking at activations

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ DE GENÈVE

An alternative approach is to look at the activations themselves.

Since the convolutional layers maintain the 2d structure of the signal, the activations can be visualized as images, where the local coding at any location of an activation map is associated to the original content at that same location.

Given the large number of channels, we have to pick a few at random.

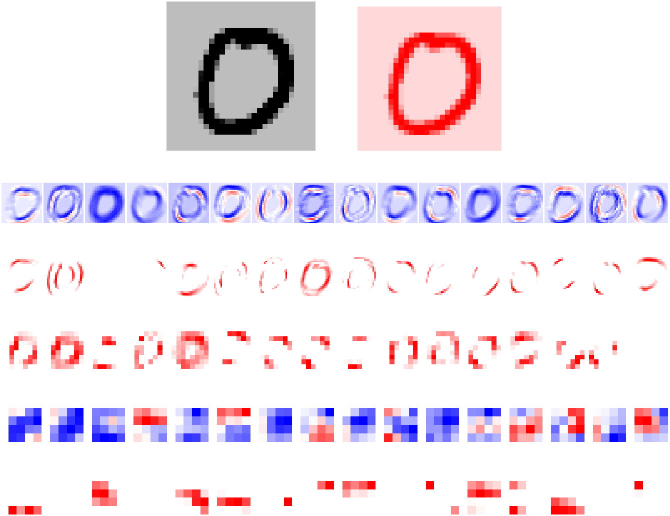
An alternative approach is to look at the activations themselves.

Since the convolutional layers maintain the 2d structure of the signal, the activations can be visualized as images, where the local coding at any location of an activation map is associated to the original content at that same location.

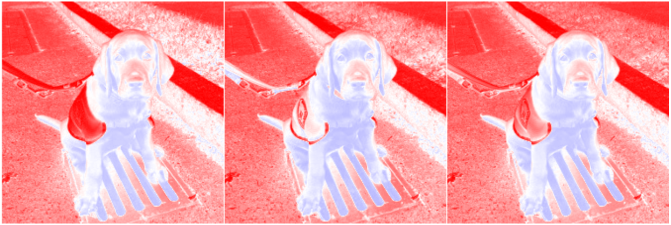
Given the large number of channels, we have to pick a few at random.

Since the representation is distributed across multiple channels, individual channel have usually no clear semantic.

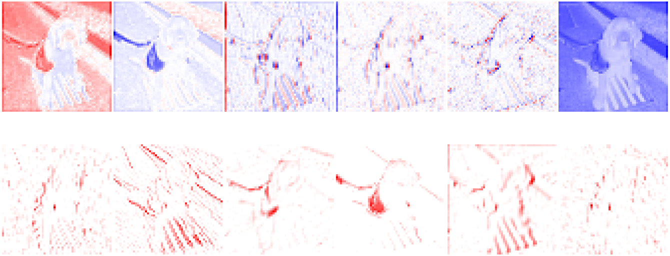
A MNIST character with LeNet (LeCun et al., 1998).



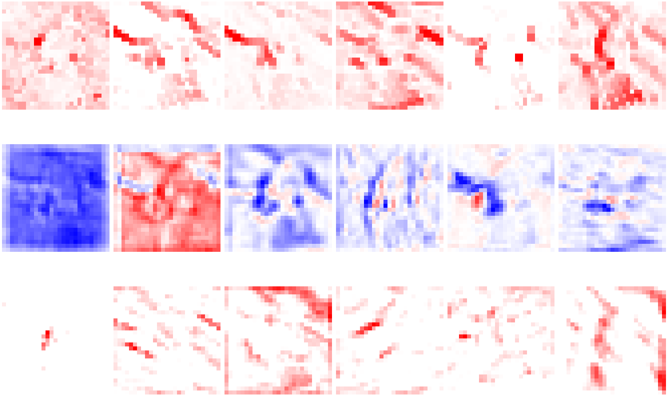
An RGB image with AlexNet (Krizhevsky et al., 2012).



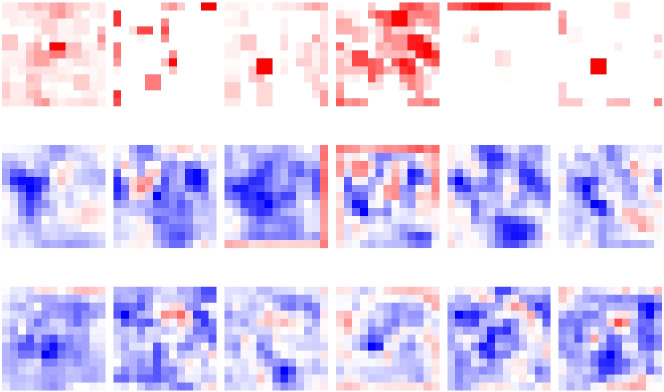
An RGB image with AlexNet (Krizhevsky et al., 2012).



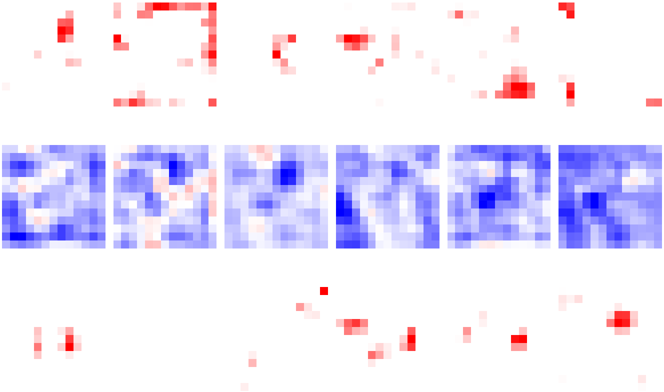
An RGB image with AlexNet (Krizhevsky et al., 2012).



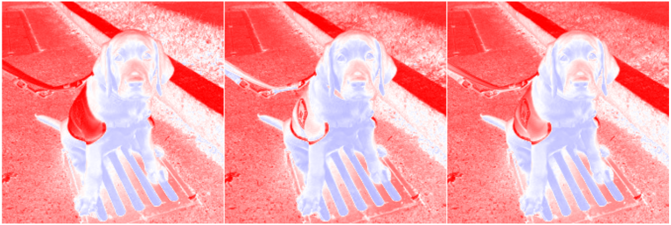
An RGB image with AlexNet (Krizhevsky et al., 2012).



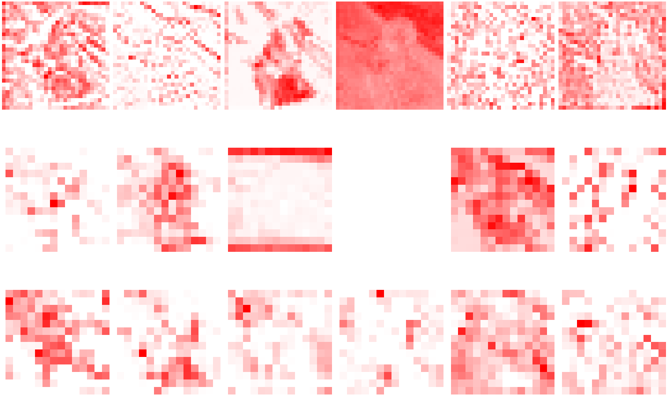
An RGB image with AlexNet (Krizhevsky et al., 2012).



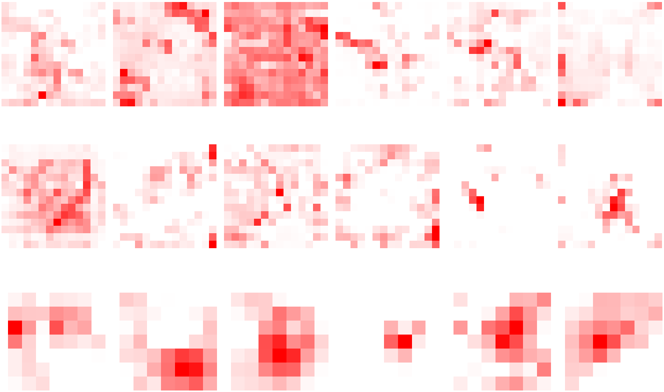
ILSVRC12 with ResNet152 (He et al., 2015).



ILSVRC12 with ResNet152 (He et al., 2015).



ILSVRC12 with ResNet152 (He et al., 2015).



Yosinski et al. (2015) developed analysis tools to visit a network and look at the internal activations for a given input signal.

This allowed them in particular to find units with a clear semantic in an AlexNet-like network trained on ImageNet.

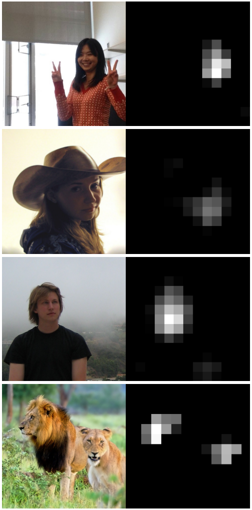
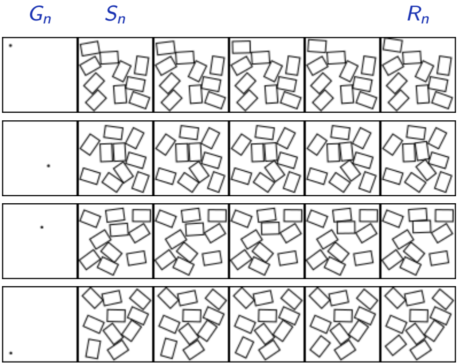


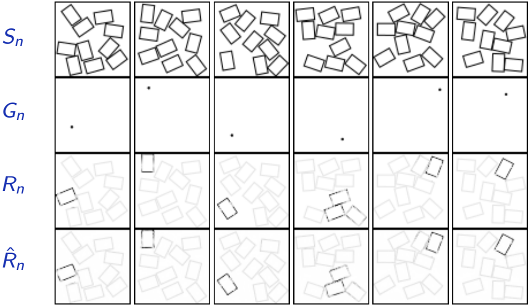
Figure 2. A view of the 13×13 activations of the 151^{st} channel on the conv5 layer of a deep neural network trained on ImageNet, a dataset that does not contain a face class, but does contain many images with faces. The channel responds to human and animal faces and is robust to changes in scale, pose, lighting, and context, which can be discerned by a user by actively changing the scene in front of a webcam or by loading static images (e.g. of the lions) and seeing the corresponding response of the unit. Photo of lions via Flickr user arnolouise, licensed under CC BY-NC-SA 2.0.

(Yosinski et al., 2015)

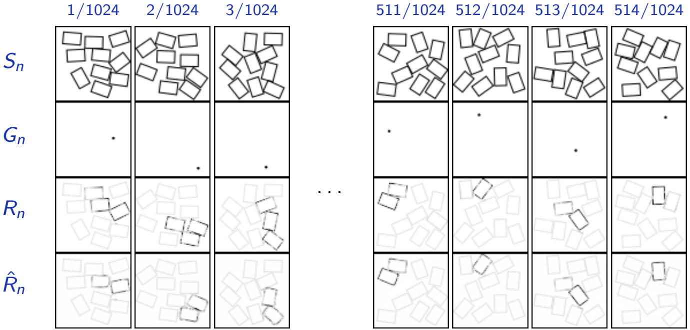
Prediction of 2d dynamics with a 18 layer residual network.



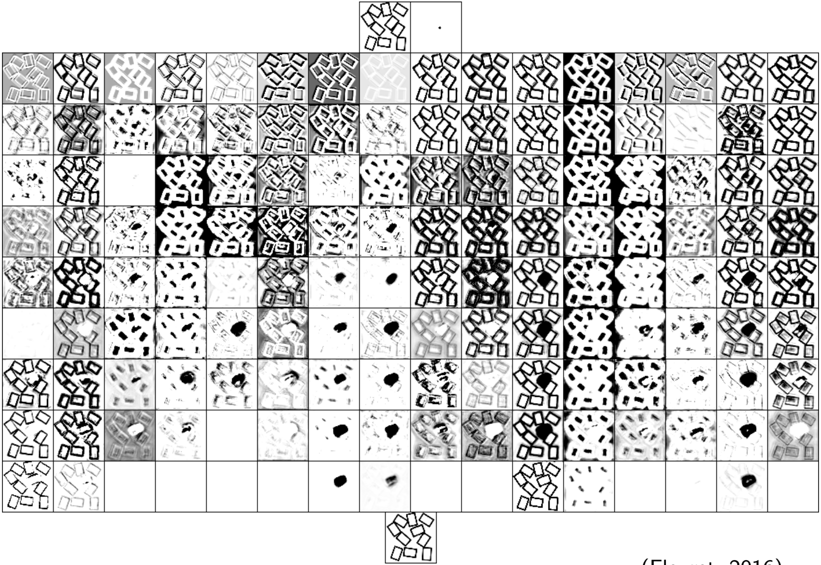
(Fleuret, 2016)



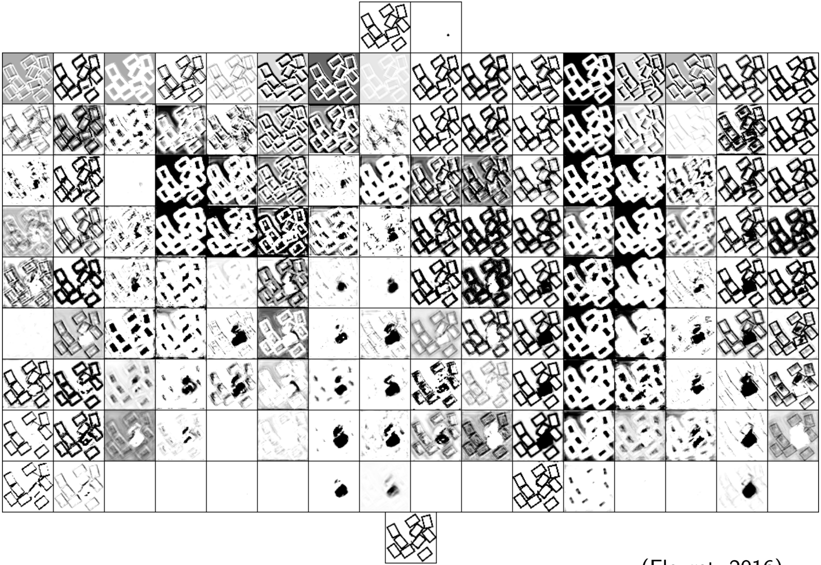
(Fleuret, 2016)



(Fleuret, 2016)



(Fleuret, 2016)



(Fleuret, 2016)

Layers as embeddings

In the classification case, the network can be seen as a series of processings aiming as disentangling classes to make them easily separable for the final decision.

In this perspective, it makes sense to look at how the samples are distributed spatially after each layer.

The main issue to do so is the dimensionality of the signal. If we look at the total number of dimensions in each layer:

- A MNIST sample in a LeNet goes from 784 to up to 18k dimensions,
- A ILSVRC12 sample in ResNet152 goes from 150k to up to 800k dimensions.

This requires a mean to project a [very] high dimension point cloud into a 2d or 3d “human-brain accessible” representation

We have already seen PCA and k-means as two standard methods for dimension reduction, but they poorly convey the structure of a smooth low-dimension and non-flat manifold.

We have already seen PCA and k-means as two standard methods for dimension reduction, but they poorly convey the structure of a smooth low-dimension and non-flat manifold.

It exists a plethora of methods that aim at reflecting in low-dimension the structure of data points in high dimension.

Given data-points in high dimension

$$\mathscr{D} = \left\{ x_n \in \mathbb{R}^D,\ n = 1, \dots, N \right\}$$

the objective of data-visualization is to find a set of corresponding low-dimension points

$$\mathscr{E} = \left\{ y_n \in \mathbb{R}^C,\ n = 1, \dots, N \right\}$$

such that the positions of the ys “reflect” that of the xs.

The **t-Distributed Stochastic Neighbor Embedding** (t-SNE) proposed by van der Maaten and Hinton (2008) optimizes with SGD the y_is so that the distributions of distances to close neighbors of each point are preserved.

The **t-Distributed Stochastic Neighbor Embedding** (t-SNE) proposed by van der Maaten and Hinton (2008) optimizes with SGD the y_is so that the distributions of distances to close neighbors of each point are preserved.

It actually matches for $\mathbb{D}_{\mathsf{KL}}$ two distance-dependent distributions: Gaussian in the original space, and Student t-distribution in the low-dimension one.

The scikit-learn toolbox

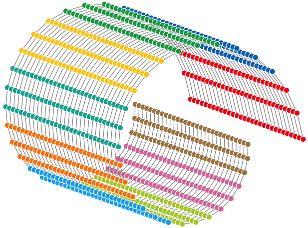
http://scikit-learn.org/

is built around SciPy, and provides many machine learning algorithms, in particular embeddings, among which an implementation of t-SNE.

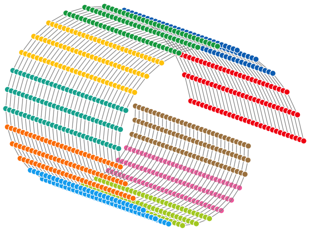
The only catch to use it in PyTorch is the conversions to and from numpy arrays.

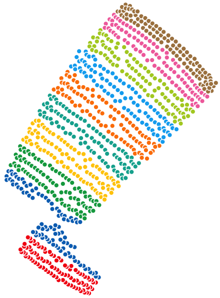
```
from sklearn.manifold import TSNE

# x is the array of the original high-dimension points
x_np = x.numpy()
y_np = TSNE(n_components = 2, perplexity = 50).fit_transform(x_np)
# y is the array of corresponding low-dimension points
y = torch.from_numpy(y_np)
```

`n_components` specifies the embedding dimension and `perplexity` states [crudely] how many points are considered neighbors of each point.



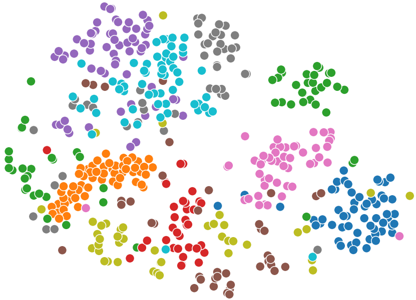
t-SNE unrolling of the swiss roll (with one noise dimension)





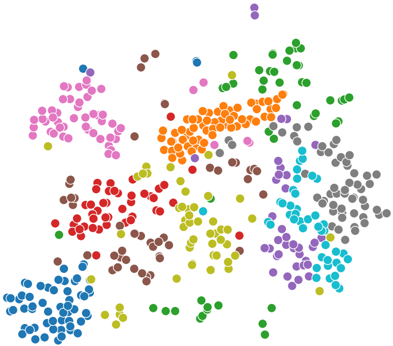
t-SNE unrolling of the swiss roll (with one noise dimension)

Input



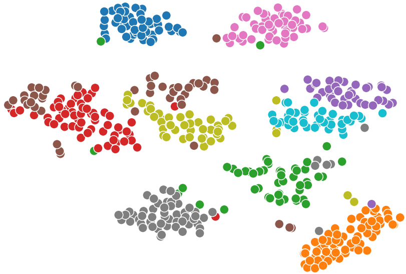
t-SNE for LeNet on MNIST

Layer 1



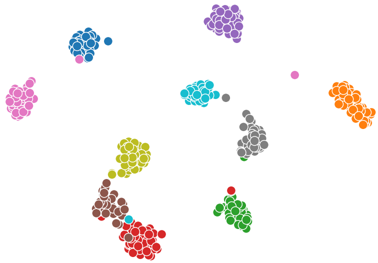
t-SNE for LeNet on MNIST

Layer 4



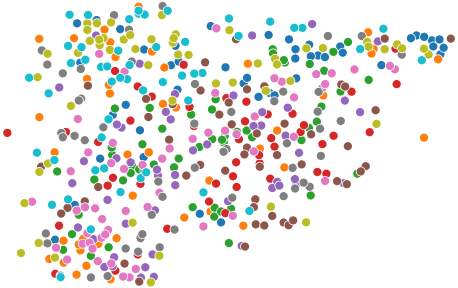
t-SNE for LeNet on MNIST

Layer 7

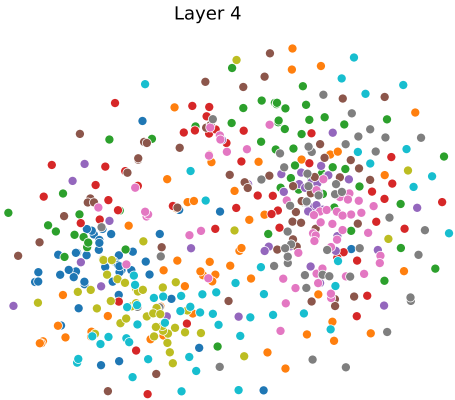


t-SNE for LeNet on MNIST

Input

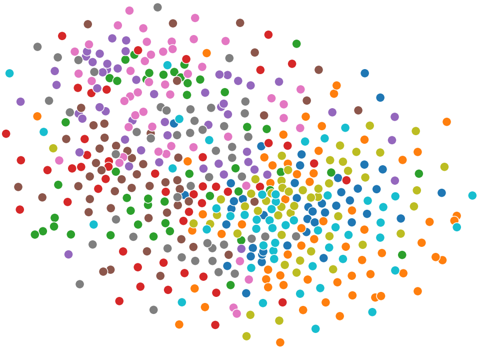


t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10



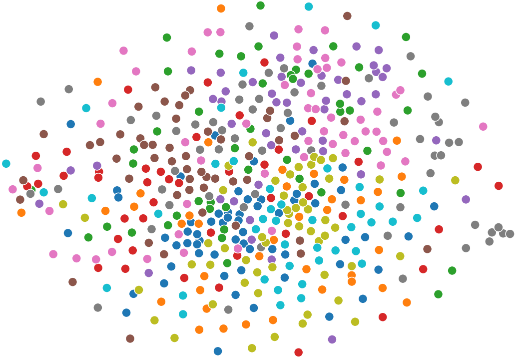
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 14



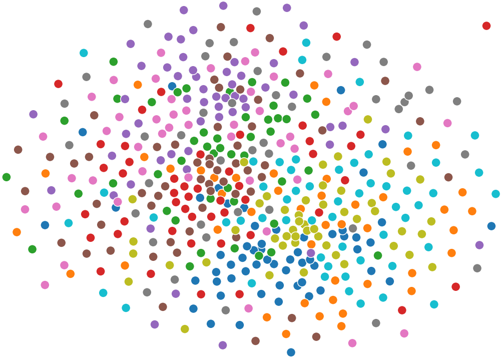
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 24



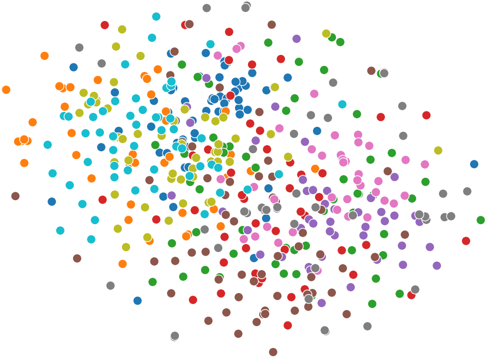
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 34



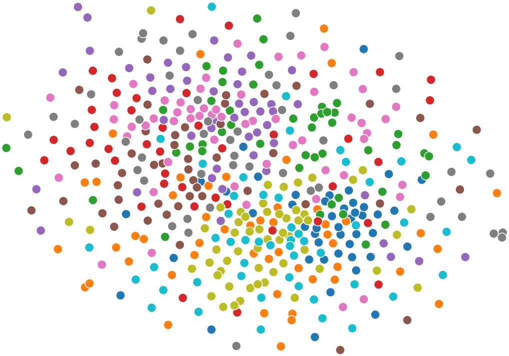
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 44



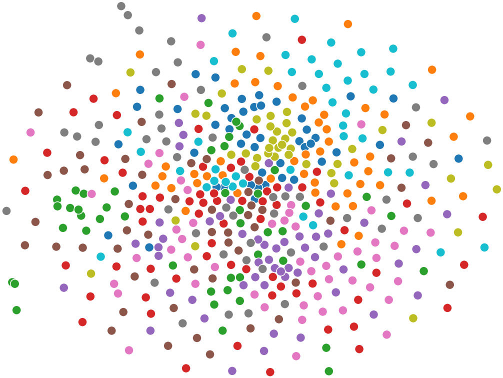
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 54



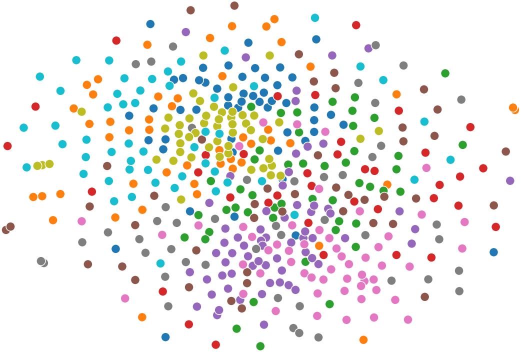
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 56



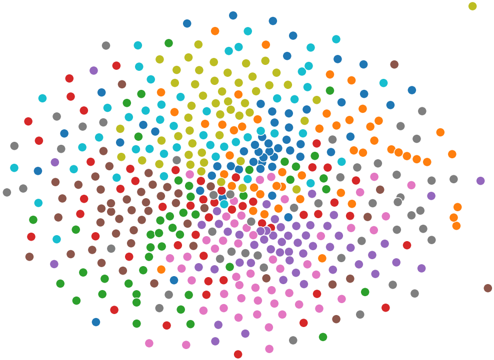
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 58



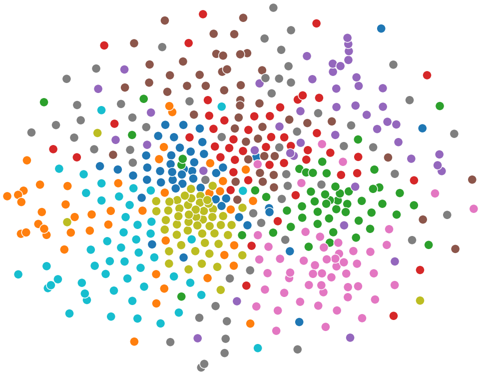
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 60



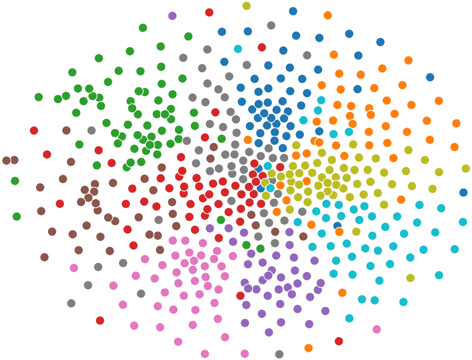
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 62



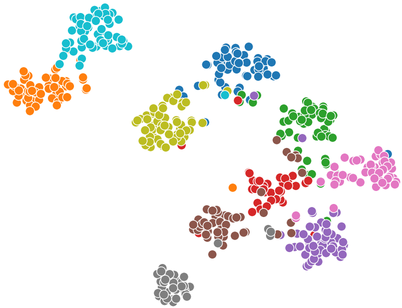
t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 64



t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

Layer 65



t-SNE for a home-baked ResNet (no pooling, 66 layers) CIFAR10

The end

References

F. Fleuret. **Predicting the dynamics of 2d objects with a deep residual network**. CoRR, abs/1610.04032, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. **Deep residual learning for image recognition**. CoRR, abs/1512.03385, 2015.

A. Krizhevsky, I. Sutskever, and G. Hinton. **Imagenet classification with deep convolutional neural networks**. In Neural Information Processing Systems (NIPS), 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 86(11):2278–2324, 1998.

L. van der Maaten and G. Hinton. **Visualizing high-dimensional data using t-SNE**. Journal of Machine Learning Research (JMLR), 9:2579–2605, 2008.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. **Understanding neural networks through deep visualization**. In Deep Learning Workshop, International Conference on Machine Learning (WS/ICML), 2015.