Self-Organizing Representation Learning

N. Kermiche

***Abstract*— Using data augmentation techniques, unsupervised representation learning methods extract features from data by training artificial neural networks to recognize that different views of an object are just different instances of the same object. We extend current unsupervised representation learning methods to networks that can self-organize data representations into two-dimensional (2D) maps. The proposed method combines ideas from Kohonen's original self-organizing maps (SOM) and recent development in unsupervised representation learning. A ResNet backbone with an added 2D *Softmax* output layer is used to organize the data representations. A new loss function with linear complexity is proposed to enforce SOM requirements of winner-take-all (WTA) and competition between neurons while explicitly avoiding collapse into trivial solutions. We show that enforcing SOM topological neighborhood requirement can be achieved by a fixed radial convolution at the 2D output layer without having to resort to actual radial activation functions which prevented the original SOM algorithm from being extended to nowadays neural network architectures. We demonstrate that when combined with data augmentation techniques, self-organization is a simple emergent property of the 2D output layer because of neighborhood recruitment combined with WTA competition between neurons. The proposed methodology is demonstrated on SVHN and CIFAR10 data sets. The proposed algorithm is the first end-to-end unsupervised learning method that combines data self-organization and visualization as integral parts of unsupervised representation learning.**

***Index Terms*— Self-Organizing Maps, Data Visualization, Unsupervised Representation Learning, SOM.**

I. Introduction

Kohonen's seminal work on self-organizing maps (SOM) [1] was an attempt to model the brain's organization into topological maps. Kohonen's theory showed that these maps can be organized using simple topological neighborhood recruitment and winner-take-all (WTA) mechanisms.

Discoveries in biological neural networks such as grid and place cells showed that topological features in the brain extend beyond sensory and motor maps to topological strategies for navigation in the environment. Emerging evidence suggests that the brain could also be encoding abstract knowledge in a similar topological way [2, 27, 34]. And while evidence for the brain's topological features has gained strength through the years, Kohonen's ideas did not keep pace with modern artificial neural network developments.

The original SOM was found to not optimize any particular loss function [3], and despite improvements in subsequent versions of SOM with well-defined clustering and vector quantization loss functions [25], learning in multi-layered architectures with SOM outputs was proven difficult because gradient-based optimization method, which is the workhorse of modern artificial neural networks, cannot be used in SOM because of the lack of easily differentiable loss functions due to SOM's reliance on non-differentiable WTA mechanisms and radial activation functions. Autoencoders were proposed as an alternate way to generate SOM [35]. However, these methods shifted the original difficulties from the neural network output layer to another layer but did not resolve the hard-WTA and radial activation problems.

Modern artificial neural networks have moved away from Kohonen's ideas. However, visualization techniques such as t-SNE [28] show that modern networks do perform some form of self-organization is some high dimensional space as a by-product of learning from data [29,32]. And while the use of *Softmax* activation can be viewed as a form of the WTA mechanism, *Softmax* WTA response is just another by-product of one-way classification. Convolutions and pooling operations in neural networks do use neighborhood pixel and neural information during feature extraction in the input and hidden layers, but topological structure is abandoned in the output layer of the network where the network's final decisions are made.

The goal of machine learning is to design systems that can accumulate knowledge by unsupervised interaction with the environment without data labelling and annotation. To achieve that goal, current unsupervised representation learning, and self-supervised learning methods, extract features from visual data by training artificial neural networks to recognize that different views of an object are just different instances of the same object. Learning is achieved by maximizing agreement between these views using metrics such as cosine-based similarity between unit-vector representations in some high dimensional embedding space. Based on this idea, various methods were proposed to demonstrate that neural networks are fully capable of extracting features from data without using prior data labeling. However, while all the various methods apply similar data augmentation techniques to generate the multiple views, the proposed methods differ when answering the fundamental question that arises when implementing unsupervised methods: how can learning algorithms be

Paper was first submitted on 10/17/2021.

TensorFlow scripts will be provided upon request.

N. Kermiche is with Western Digital Corporation, Irvine, CA 92612 (e-mail: noureddine.kermiche@wdc.com)

prevented from collapsing into trivial solutions where representations of all objects are the same?

To prevent collapse, some of the proposed methods rely on contrastive learning where augmented versions of the same image instances ("positives") are contrasted with instances extracted from other images ("negatives") [4,5,6,8,9,22]. A loss objective based on NCE transformation of the cosine similarity prevents collapse in an explicit manner by encouraging the representations for positive pairs to be close while representations for negative pairs to be far apart. Contrastive methods have a large memory footprint because of the quadratic complexity in the batch size. Other methods use only the "positive" pairs and rely on simple cosine similarity loss with linear complexity in the batch size [13,24]. To prevent collapse, these methods use some combination of asymmetric learning on dual-network architectures and stop-gradient operations. The experimental results achieved by these methods are as powerful as the contrastive methods, but they do not offer explicit explanations as to how avoiding collapse is achieved. Because the issue is still an active research subject, there are probably more methods for preventing collapse than we can account for or categorize in this study such as: explicit collapse avoidance using correlation-based whitening algorithms with quadratic complexity in the embedding vector length [23,26,33].

The methods presented so far are called self-supervised learning methods because of their need to use labels. The methods are two-phased processes where features are extracted first using unsupervised learning before a classifier is constructed in the second phase. The second phase requires supervised learning on a limited set of samples because there is no way of quantifying how good the features are without going back to the labels when the original intention was to avoid using in the first place. But despite using a limited number of labels, these methods proved the validity of the unsupervised learning phase by achieving state-of-the-art results in many challenging classification tasks.

Using labels is completely avoided with recent advances in end-to-end deep clustering using unsupervised representation learning [15, 29, 32]. The losses used are similar to the methods presented previously but the losses are applied directly at the level of the clusters with added cluster assignment balancing algorithms such as the Sinkhorn-Knop transform [15]. These otherwise powerful algorithms have two glaring weaknesses when striving for end-to-end unsupervised learning. The first is that the desired number of clusters must be defined a priory. The second is that these methods use separate visualization techniques such as t-SNE [28] to confirm the quality of the clusters and demonstrate the topological relations between cluster assignments. Relying on visualization techniques that are not part of the end-to-end unsupervised learning process and forcing a desired number of clusters are the main motivations behind this study.

This study proposes end-to-end unsupervised representation learning where self-organization and visualization are integrated into the learning process. We borrow the idea of neural selectivity seen in biological neurons to introduce a new approach where each neuron in a large two-dimensional (2D) output layer is its own cluster. The proposed algorithm is called URL-SOM because it combines recent advances in unsupervised representation learning with ideas from Kohonen's original SOM.

In addition to the linear complexity of the cosine similarity loss used in representation learning, we propose a new loss function with linear complexity that enforces SOM requirements of WTA and competition between neurons while explicitly avoiding collapse into trivial solutions. We also show that SOM topological neighborhood requirements can be achieved using a fixed convolution in a 2D output layer without resorting to radial activation functions which prevented the original SOM algorithm from being extended to nowadays neural network architectures. A tentative idea of using 1x1 convolutions to integrate representations in a neural network was proposed in [30]. Our proposed method uses a fixed radial convolution where the support of the filter is a free hyperparameter that is optimized based on the SOM task at hand as we will see in the experimental section of this study.

In Section II, the contributions of this study are highlighted. In Section III, we introduce the network architecture used in this study. In Section IV, we propose a new loss function. In Section V, the proposed methodology is demonstrated on SVHN and CIFAR10 data sets using a ResNet backbone.

II. CONTRIBUTION

This study proposes end-to-end unsupervised representation learning where self-organization and visualization are integrated into the learning process. Based on a simple geometric interplay between *Softmax* and L2 normalization, we propose a new loss function with linear complexity that enforces SOM requirements of WTA and neuron competition while explicitly avoiding collapse into trivial solutions. We show that SOM topological neighborhood requirements can be achieved with a fixed convolution in the projection layer without resorting to radial activation functions.

III. THE NETWORK

A. URL_SOM network

As shown in Fig. 1, URL-SOM network is similar to networks used for classification tasks except that the number of neurons in the output layer could be very large to be able to form a 2D map and a fixed convolution is added just before the final *Softmax* output.

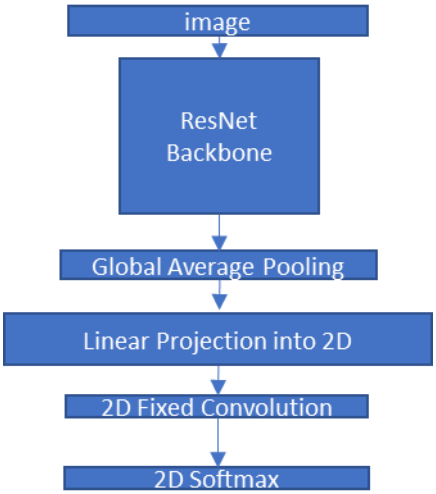

Fig1. URL_SOM network. The projection, convolution and *Softmax* are 2D single channel layers. Notice the fixed 2D Gaussian convolution after the linear projection and before the Softmax. Adding dropout or L2 regularization to linear projection layer (not shown in the figure) improved SOM response slightly.

The backbone used in this study is a ResNet. A standard global average pooling is used before a linear projection is converted into a 2D layer. After that, a fixed convolution (see next section) is used before the final *Softmax* layer.

We tried to build the 2D layer by utilizing incremental deconvolution operations like what is done in GANs when constructing 2D outputs. Deconvolution could be less expensive than a large linear projection. However, we did not see drastic improvements and decided to stay with the linear projection since we have less architectural parameters to tune. With a linear layer, the size of the 2D projection is the only tunable parameter.

B. *Topological neighborhood using 2D fixed radial filter*

We tried many strategies to enforce SOM topological neighborhood requirement by attempting to smooth the 2D response and could not find anything that worked better with the proposed loss function than a simple convolution with a fixed radial filter.

One of the failed strategies was to minimize the error between the 2D output and radial targets that are generated recursively during the learning process based on winning neurons. This approach would have been the closest to the original SOM with hard-WTA mechanism. The approach did work for MNIST and FASHION-MNIST data sets when using shallow networks but did no work with deeper networks and more challenging data sets such as CIRAF10.

As shown in Fig. 1, we found that applying the fixed 2D convolution directly after the 2D linear projection is crucial. Introducing any nonlinear activations between the two linear operations degraded the SOM quality.

Inspired by t-SNE [28], we also tried to see if the 'crowding phenomenon" seen when visualizing data using Gaussian embedding distribution compared with t-student distribution can be seen in our proposed learning algorithm when we switch from a Gaussian filter to a t-student filter. The SOM qualitative response improved slightly if we use t-student filters as shown in the experimental section of this study, but we did not see the drastic change seen in t-SNE. A simple average pooling was also tried. However, the optimization became too difficult when using simple averaging and undesirable SOM artifacts were seen with simple averaging. We also found that allowing the filter to be learned instead of using a fixed filter resulted in poor SOM.

The size of the fixed Gaussian kernel is set to 6-sigma instead of the standard 3-sigma used for Gaussian image blurring. Artifacts were seen in the final SOM if we reduce the kernel size.

IV. SOM LOSS

A. *Softmax, L2 normalization and cosine similarity*

The cosine similarity is a crucial component when optimizing a loss objective in many current unsupervised representation learning methods [7,13,14,24,29]. This is usually implemented with an L2 normalization of the latent-space representations which corresponds to projecting the features on the surface of the unit hypersphere.

Our proposed loss will be based on similar cosine similarity measures. To achieve that, a *Softmax* output vector $p = \{p_i\}_1^n$ give the (flattened) *n* activations $\{v_i\}_1^n$ after the 2D convolution is:

$$p_i = \frac{exp(v_i)}{\sum_{j=1}^{n} exp(v_j)} \quad i = 1, \dots n.$$

Vector *p* satisfies the probability constraints:

$$\sum_{i=1}^{n} p_i = 1 \ and \ p_i \geq 0, \tag{1}$$

which defines a hyperplane in an *n* dimensional space and is projected into a unit hypersphere using

$$P = \frac{p}{\|p\|} \tag{2}$$

where

$$\|p\| = \left(\sum_{i-1}^{n} p_i^2\right)^{\frac{1}{2}}$$

In the following section, a vector *p* and point with the same name *p* are used interchangeably. Vector *p* starts at the origin *O*, where all probabilities are zero, and ends at the corresponding point *p* (see Fig. 2).

The L2 projection of the *Softmax* hyperplane is a section of the unit hypersphere for positive probabilities. A special constant vector defined by point u where *Softmax* has equal probabilities $p_i = \frac{1}{n} \ for \ i = 1, \dots n$ and has a constant L2 projection vector *U*. Fig. 2 illustrates u and *U* vectors and the *Softmax* line between p_1 and p_2 and the unit circle for $n = 2$. Vector *U* is crucial to the definition of the proposed loss.

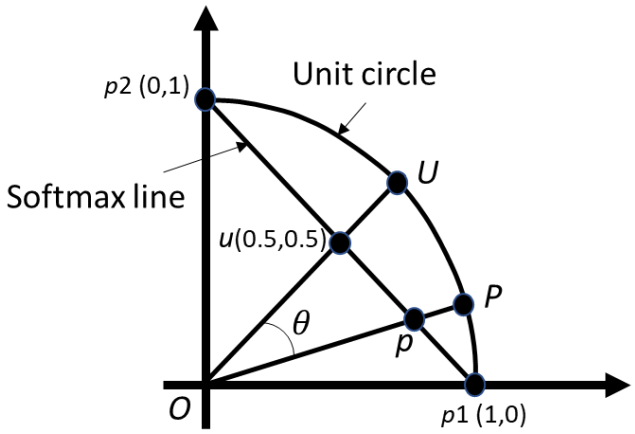

Fig. 2. Softmax output vector *p* is on the line between 2 WTA vectors *p*1 and *p*2. L2 normalization will take p from the Softmax line into *P* on the unit circle. Equal probability vector *u* is transformed into *U*.

On the unit hypersphere, the vector scalar product between vectors *P* and *U* is a true cosine similarity metric since

$$P.U = \sum_{i=1}^{n} p_i u_i = \cos(\theta) \ , \tag{3}$$

where θ is the angle between *U* and *P*. The definition (3) of the vector scalar product is used for the remainder of this section.

B. Winner-take-all loss

Winner-take-all (WTA) points are corners of a hypercube. In Fig. 2, *p1* and *p2* correspond to WTA vectors for $n = 2$. Enforcing WTA response is nothing other than pushing vector *P* away from *U* on the hypersphere towards the corners of the hypercube. If we are given *N* vectors in a batch $\{P_i\}_1^N$, this can be achieved by increasing the angles between the *N* vectors and the *U* vector or minimizing the following loss

$$loss_{WTA} = \frac{\sum_1^N P_i}{N} \ . U \tag{4}$$

where *U* is the equal-probability constant unit vector defined previously. Loss (4) is a positive metric because are probabilities are positive. WTA loss has linear complexity because it is computed based on the averaging operation through a batch.

C. Loss objective for preventing collapse

Minimizing $loss_{WTA}$ could cause all data representation vectors to move towards very few corners of the hypercube resulting in a collapse of the *Softmax* output. To prevent that, we are going to enforce a symmetry condition where the angle between the average vector $a = \frac{\sum_1^N P_i}{N}$ in a batch of size *N* and vector *U* is minimized. To achieve that, we project vector *a* into the unit hypersphere vector $A = \frac{a}{\|a\|}$. In Fig. 2, if the network collapses into a single WTA (either *p1* or *p2*), the angle formed between the average response vector over a batch *A* and vector *U* will be maximum. The only way for the average vector direction to align with *U* is if *p1* is WTA for 50% of the samples in the batch and *p2* is WTA for the other 50%. The symmetry relation represented by vector *U* forces WTA for the various samples to fill the hypercube corners evenly and the relation can be extended to any dimension *n*.

To minimize the angle between *A* and *U*, the cosine similarity between *A* and *U* must be maximized. Preventing collapse is achieved by minimizing the positive metric

$$loss_{collapse} = 1 - A \ . U \tag{5}$$

Preventing collapse loss also has linear complexity because it is computed based on the averaging operation through a batch.

The geometric and statistical arguments presented so far could imply that the batch size must be a lot larger than the number of neurons in the *Softmax* output for the average operation to give valid statistics when using loss (5). However, all simulation results show that we could use $N = 128$ and $N = 256$ with *Softmax* size *n* being in the thousands without seeing noticeable degradation in SOM behavior. We do not have an explanation as to why we could get away with using small batch sizes except that it may be another positive side effect of stochastic gradient learning.

Loss (5) is deceptively simple. Whereas $loss_{WTA}$ (4) is trying to push representations towards the *n* corners of the hypercube, $loss_{collapse}$ is trying to distribute representations evenly among the *n* output neurons. A simulation of the $loss_{WTA} + loss_{collapse}$ for $n = 2$ is shown in Fig. 3. The *Softmax* vectors are constructed using the logistic function with a single input $x \geq 0$:

$$p(x) = \left[\frac{1}{1 + \exp(-x)}, \frac{\exp(-x)}{1 + \exp(-x)}\right]$$

We vary the *Softmax* input *x* from 0 to 6. We use a binomial probability for selecting a point *p(x)* (close to *p*1) or (1 - *p(x)*) (close to *p*2). The probability was changed from 0 to 1. The plot shows clearly, as predicted previously, that the minimum is reached when the *Softmax* is saturated for large inputs (WTA) and the probability of selecting *p*1 is 50%. The idea was justified by a symmetry argument that should hold for any n. However, to perform similar simulations to $n = 2$ to illustrate that the idea still works for large n requires a more challenging Monte Carlo simulation. Further, the idea will be confirmed with real data and a large n in the following experimental section of this study.

$loss_{collapse}$ prevents collapse in an explicit manner by trying to distribute representations evenly among the output neurons without resorting to cluster assignment balancing algorithms such as the Sinkhorn-Knop transform [15], where the ratio between the batch size and the desired number of clusters must be defined. In our case, we do not need to define the desired

number of clusters because each neuron in the 2D output layer is its own cluster.

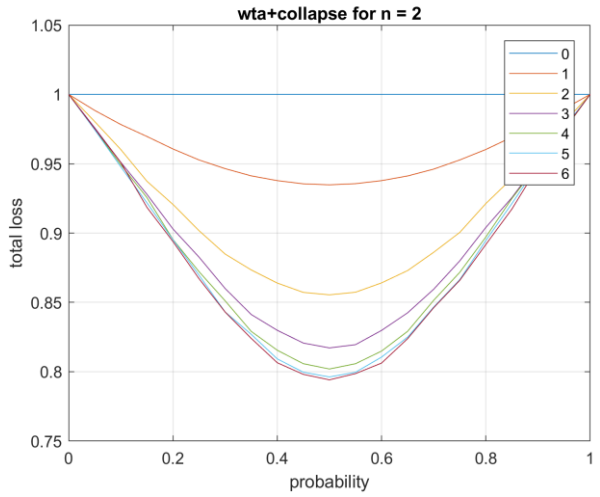

Fig. 3. We vary the *Softmax* input from 0 to 6. The minimum of the combined loss $loss_{WTA} + loss_{collapse}$ is reached when *Softmax* is saturated for large inputs (WTA) and the probability of selecting $p1$ is 50%.

D. Similarity and combined loss

The similarity loss is like the loss used elsewhere [7,13,14,24,29]. Two instances of an image are generated using random image transformations such as cropping, brightness change, left/right flipping and other transformations [22]. Given two instances of the same image, we get two *Softmax* outputs $p^+ and\ p^-$. The *Softmax* vectors are projected into unit vectors $P^+ and\ P^-$. The similarity between N sample pairs in a batch is

$$loss_{similarity} = 1 - \frac{1}{N}\sum_{1}^{N} P_i^+ . P_i^- \qquad (6)$$

When given N image pairs, $loss_{WTA}$ and $loss_{collapse}$ must be calculated for $2N$ samples. We used the following N average samples:

$$P_i = \frac{P_i^+ + P_i^-}{2}, i = 1, \dots, N \qquad (7)$$

to calculate $loss_{WTA}$ and $loss_{collapse}$ using (4) and (5) directly. The total SOM loss is

$$loss = loss_{WTA} + loss_{collapse} + \gamma\, loss_{similarity} \qquad (8)$$

and $\gamma > 0$ is an added weight factor. Simulation results show that $loss_{WTA}$ and $loss_{collapse}$ must be scaled similarly as shown in (8). γ can be tuned based on the sharpness and the number of clusters seen in the final SOM. The SOM sensitivity, however, does not vary a lot as we change γ between 1 and 2.

Since $loss_{WTA}$ and $loss_{collapse}$ have linear complexity in both batch size and output size, the total loss, which is based on 3 cosine similarities, also has linear complexity in both batch size and output size because of the linear complexity of $loss_{similarity}$.

Since optimization is performed by the projection of the *Softmax* output on the hypersphere, the reader may wonder why we need the *Softmax* layer in the first place. Losses (4) and (5) won't be possible without positivity conditions on the probabilities. *Softmax* is one way to use a differentiable transformation that preserves probability conditions shown in (1).

The following script shows a TensorFlow implementation of the total loss function.

```
const_vectorU= tf.ones([1,SOM_Size*SOM_Size])
const_vectorU = tf.math.l2_normalize(const_vector, axis=1)

def compute_loss(p, z):

  # Note that here we are enforcing the network to match
  # the representations of two differently augmented batches p and z
  # of data and also prevent collapse and encourage winner-take-all response

  # L2 normalization
  p_l2 = tf.math.l2_normalize(p, axis=1)
  z_l2 = tf.math.l2_normalize(z, axis=1)

  # Avarage vector
  p_z_mean = tf.reduce_mean((p_l2+z_l2)/2, axis=0, keepdims=True)

  #L2 normalization of average vector
  p_z_mean_l2 = tf.math.l2_normalize(p_z_mean, axis=1)

  #loss to prevent collapse
  cos_similarity = tf.reduce_sum(tf.multiply(p_z_mean_l2, const_vectorU), axis=1)
  loss_collapse = 1 - tf.reduce_mean(cos_similarity)

  #WTA loss
  cos_similarity = tf.reduce_sum(tf.multiply(p_z_mean, const_vectorU), axis=1)
  loss_wta = tf.reduce_mean(cos_similarity)

  #Similarity loss between 2 augmented views
  cos_similarity = tf.reduce_sum(tf.multiply(p_l2, z_l2), axis=1)
  loss_similarity = 1 - tf.reduce_mean(cos_similarity)

  #total loss
  gamma = 2
  loss = loss_collapse + loss_wta + gamma*loss_similarity

  return loss
```

Script: URL_SOM loss has linear complexity in batch size and output size because it is based on the sum of 3 cosine similarities.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Data Augmentation

We used the same data augmentation for both CIFAR10 and SVHN data sets (image size is 32 by 32 pixels). The image augmentation used is close to the one proposed in [22]. We used uniform random cropping with sizes between 22 to 32 pixels. The cropping transformation is applied constantly with an added random left/right flip 50% of the time. We applied random color jitter 80% of the time using random [brightness, contrast, saturation, hue] = [0.4, 0.4, 0.4, 0.4]. We also changed images to gray scale 50% of the time.

Using the combination of all these transformations as stated in [22] is crucial for good representation learning as well as the proposed SOM. We also found that SOM is sensitive to crop size and the gray scale transformation rate. We found that too much deviation from the recommended setting degraded the

SOM response.

The justification of using some specific combination of random data augmentations for representation learning is based on experimental data [22] with no theoretical framework that explains how to choose a particular image transformation given a data set combined with a given neural network architecture. Cropping could be a way to bias the network to focus on the middle of the image since most data sets have target objects centered in the middle of the image. Gray scale could be a way to ignore color and focus on geometric features. However, we cannot completely ignore the border of the image or its colors without throwing away valuable image information. It is also possible that our convolutional neural network with a smaller size (next section) and smaller data sets may not be as efficient at absorbing heavy data augmentation as larger networks such as the transformer-based network shown in [32].

B. The network, data, and training

We used the same network and training strategy for both CIFAR10 and SVHN. The network used in this study is shown in Fig. 1. The backbone is a ResNet-20 where we changed all *relu* to *elu* activations because the results obtained were superior with *elu* activation. We used a ResNet-20 with 16, 48, and 256 (3x3) filters for the 3 stages. The standard ResNet global average pooling was implemented as an average pooling layer with pool_size = 8. The pooling is followed by a linear projection into a 30x30 2D layer (900 neurons). After that, a fixed convolution is used before the *Softmax* layer.

The network used has 3.4 million trainable parameters with 0.23 million parameters for the linear 30x30 2D projection operation. We used all training and testing data for both CIFAR10 and SVHN to train the SOM network. We trained the network for 500 epochs with early stopping if the loss does not improve after 10 epochs. We used Adam optimizer with initial warmup. The training strategy and the SOM loss as a function of epoch number for both CIFAR10 and SVHN can be seen in Fig. 5 and Fig. 10.

We found that adding regularization techniques to the SOM layer such as dropout (rate = 0.5) or L2-regularization (decay = 0.0005) improved the SOM response slightly.

C. Neural selectivity as SOM goodness metric

The network is trained without access to labels. However, we would like to measure SOM quality based on the 2D distribution of the data and see if the network did uncover topological properties in the data by clustering samples with the same labels close to each other. However, any standard clustering metric we tried to use introduced additional hyper parameters such as the number of clusters or the number of neighbors in the 2D map. To avoid that, and because our approach makes each neuron in the 2D output its own cluster (see Section *IV.C*), we used neural selectivity as a non-parametric method to measure the clustering capability of SOM and we left the 2D distribution quality to be validated through visual inspection.

Fig. 6, demonstrates how to visualize SOMs since the number of samples is larger than the number of neurons in the 2D grid. In Fig. 6, each neuron in the 2D map will have maximum response to several samples. We compute the mode of the distribution of labels and consider the label that occurs more often as the true label for that neuron. Other labels in the neuron will be considered error samples. This metric quantifies how sensitive a neuron is to a specific label. The average correct/incorrect numbers are averaged over the 2D grid. We call the metric Neural Selectivity (NS). NS is just a 0-neighbor classifier.

The NS obtained using the new SOM algorithm is 70% (Fig. 4) for CIFAR10 and 91% for SVHN (Fig. 8). 70% classification rate for CIFAR10 is an adequate result but it is not state-of-the-art for unsupervised methods. However, we believe that this is the first method that shows good end-to-end unsupervised methodology where learning and visualization are integral parts of the learning algorithm.

D. New parameters

The proposed algorithm introduces 3 new parameters: the 2D output size, the Sigma of the fixed Gaussian convolution, and the γ weight for the similarity loss. We kept the 2D size fixed to 900 (30x30) neurons. We varied Sigma from 3 to 10 and γ from 1 to 2. The effect of the changes in Sigma can be seen in Fig. 7 and Fig. 9. In general, neural selectivity improves with smaller filter support at the expense of more clusters and noisier maps. However, the change is not drastic

The effects of changing γ are not that drastic either. However, cluster size increases as we increase γ and NS improves. We used $\gamma = 2$ in this study.

E. Gaussian vs. t-student convolutional filters

In Fig. 11, CIFAR10 SOM neural selectivity with the t-student convolution filter $\frac{1}{1+(x^2+y2)}$ improved to 73.33% from 72.12% for filter $\frac{1}{1+\frac{(x^2+y2)}{18}}$ at the expense of more clusters and more dead neurons. The qualitative response and neural selectivity improved slightly from the Gaussian response if we use t-student filters with large support. However, t-student filters have larger support that Gaussian filters and require filter size that is bigger than SOM size to avoid edge artifacts.

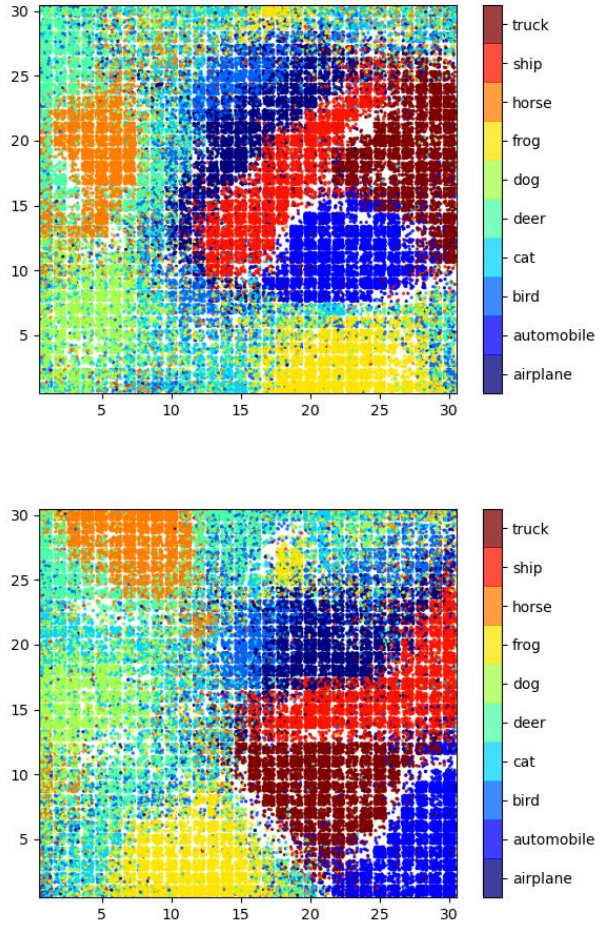

Fig. 4. CIFAR10 SOM for all 60000 samples. WTA response is shown on a 30 by 30 neuron grid after two separate runs. Fixed Gaussian filter with Sigma = 5 was used. Neural selectivity is 70.3% and 70.58%. All items have separate clusters except for the cat and the deer. Cat data surrounds the dog data, and the deer data surrounds the horse data. It is interesting that the bird and airplane data are relatively close. All non-animal objects tend to be close to each other. The two plots highlight that this is a non-linear optimization result and subsequent runs may not repeat the same exact visual distribution, but the highlighted semantic relations do persist over multiple runs. The 2D representation distribution confirms that $loss_{collapse}$ (see Section *IV.C*) is preventing collapse by encouraging representation to spread evenly among all output neurons.

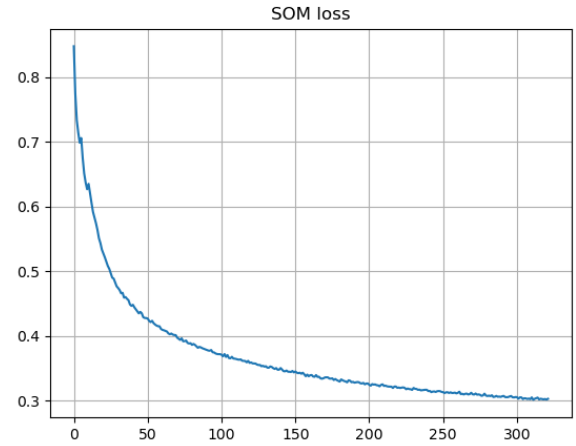

Fig. 5. CIFAR10 loss as a function of epoch number using batch size = 256. Adam optimizer with warmup is applied. Learning rate switches from 0.0001 to 0.0005 at epoch 5 and switches to 0.001 at epoch 10. This explains the two glitches seen. Learning was performed with early stopping if the loss does not improve for 10 epochs. In this case, early stopping occurred at epoch 322.

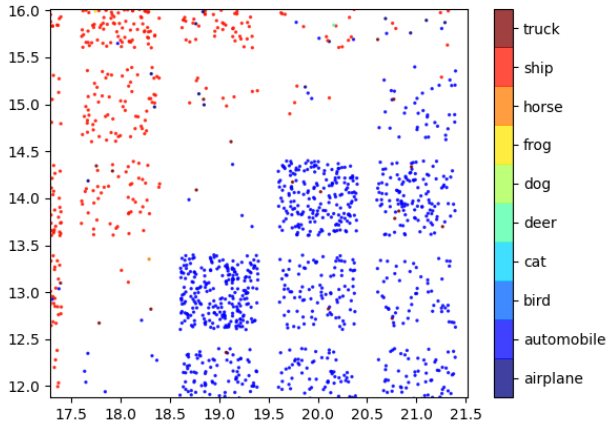

Fig. 6. SOM visualization for samples that share same winner-take-all neurons. CIFAR10 response is zoomed. Plot shows added random (-0.4 to 0.4) uniform scatter around each neuron for display purposes only to highlight all samples that share the same winner-take-all neurons. Notice that some neurons respond to fewer samples than others. Having some neurons not respond to any samples is not desirable since we would like SOM to be a space-filling response without wasting neurons. We saw in some instances that two incompatible clusters are separated by dead neurons if they are close to each other. Dead neurons could be considered a reasonable representation of discontinuity between incompatible clusters.

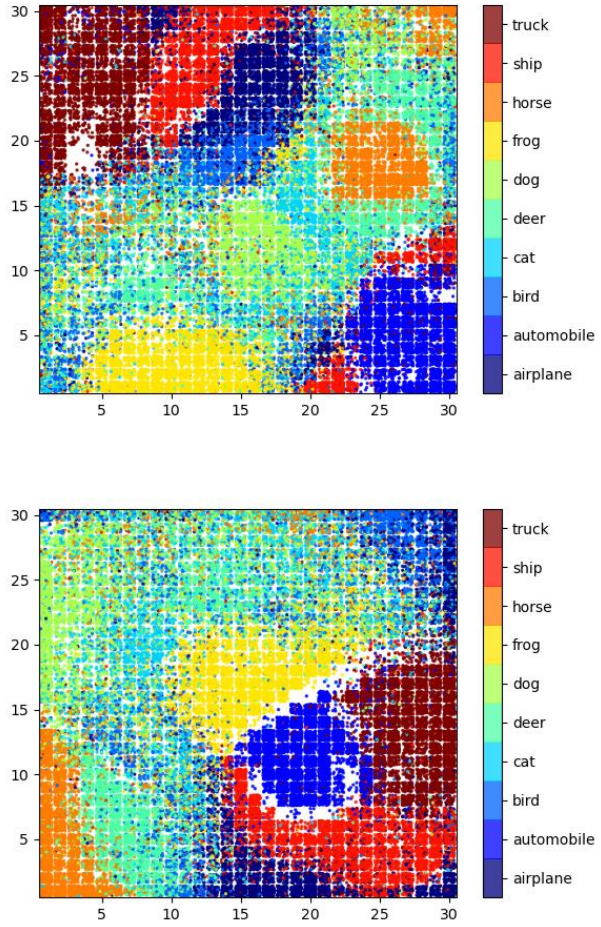

Fig. 7. CIFAR10 SOM for all 60000 samples after changing Sigma from 3 (top plot) to 10 (bottom plot). Neural selectivity decreased from to 71.19%. to 70.19%. As a general observation, neural selectivity improves with smaller filter support at the expense of more clusters and noisier maps. However, the change is not drastic. Notice the presence of non-responding neurons which is not a desirable property in terms of neural efficiency but necessary to represent discontinuity by separating incompatible clusters if they ended up close to each other.

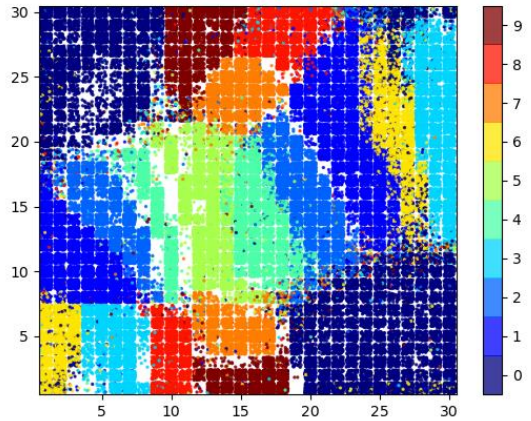

Fig. 8. SVHN SOM for all 99289 samples after 188 epochs. We used a Gaussian filter with Sigma = 5. Neural selectivity is 91.50%. All items have separate clusters. We used the same network and the same contrastive data augmentation as CIFAR10, except for left/right random image flip because it is not a valid symmetry for SVHN. There are two clusters per item representing the different street numbers. The explanation of this phenomenon is because SVHN has bright numbers with dark backgrounds and dark numbers on bright backgrounds. Numbers with opposite front/back brightness are assigned to different clusters. Neural networks are not invariant to image level inversion and trying to add image level inversion to the data augmentation reduces SOM quality.

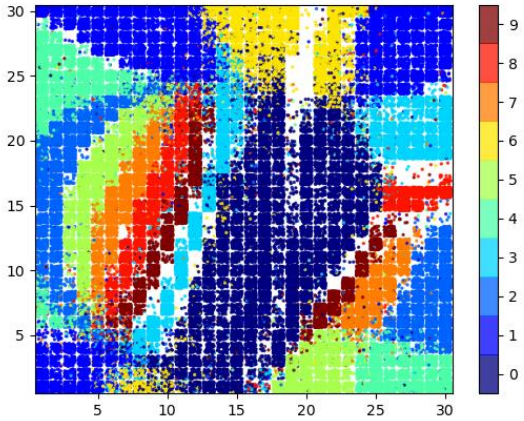

Fig. 9. SVHN SOM for all 99289 samples after increasing the Gaussian filter Sigma from 5 to 10. Neural selectivity is 91.06% which is similar to previous data with Sigma = 5. However, there are still multiple clusters per item representing the different street numbers with added artifacts not seen with Sigma = 5. Notice the lines formed by dead neurons between some clusters.

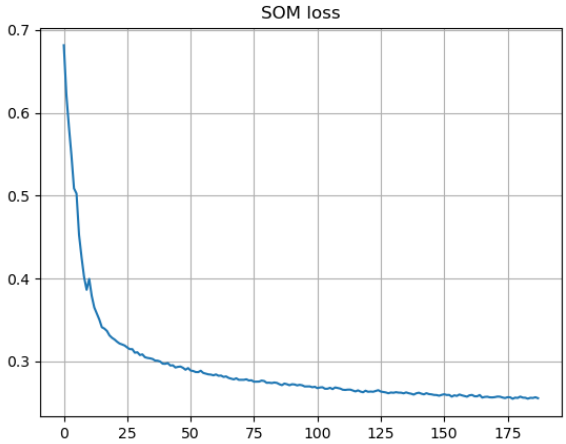

Fig. 10. SVNH loss as a function of epoch number using batch size = 256. Same optimization used as CIFAR10 but gradient descent had an early stop at epoch 188 because of loss flattening.

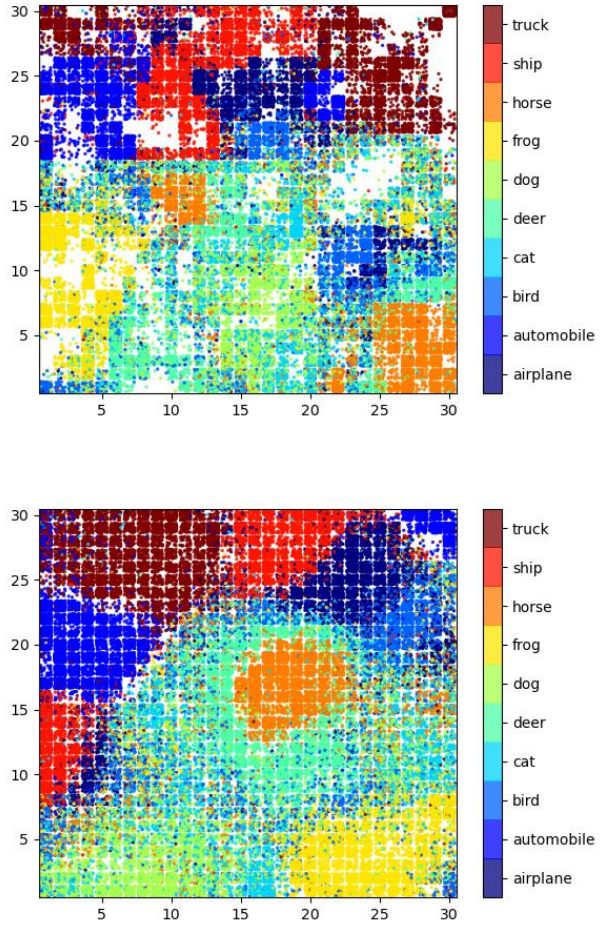

Fig. 11. CIFAR10 neural selectivity using the t-student convolution filter $\frac{1}{1+(x^2+y2)}$ in the top plot improved slightly to 73.33% from 72.12% with filter $\frac{1}{1+\frac{(x^2+y2)}{18}}$ in the bottom plot at the expense of more clusters, more dead neurons, and noisier SOM.

VI. Conclusion

This study demonstrates how to build self-organizing representation learning using gradient optimization and current neural network architectures. The proposed method combines self-organization and visualization as integral parts of machine learning without having to use external visualization techniques such as t-SNE.

We also demonstrate that we could design stable unsupervised representation learning algorithms with linear complexity using a single network by exploiting a simple geometric interplay between *Softmax* and L2 normalization.

CIFAR10 and SVHN are not vanilla sets but they are considered small sets compared to nowadays sets with millions of samples. The ResNet-20 network used in this study is also small compared with nowadays networks. Generalizing the proposed algorithm to representation maps with higher dimensions and extending the ideas presented in this paper to larger networks and larger data sets will be part of future studies.

The data presented in this study does show that some semantic relations do develop as part of representation self-organization. However, artificial neural networks are still far away from being able to encode knowledge as topological relations. This study could be one small step in that direction.

Acknowledgment

I thank my family for their support and the Western Digital Corporation for allowing me to publish this work.

References

[1] T. Kohonen. *Self-Organization and Associative Memory*. Springer Series. In Information Sciences, Vol.8. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1984.

[2] Jacob L. S. Bellmund, Peter Gärdenfors, Edvard I. Moser, Christian F. Doeller,"Navigating cognition: Spatial codes for human thinking". Science. Vol. 362, No. 64159 Nov 2018

[3] Marie Cottrell, Jean-Claude Fort, Gilles Pagès, "Theoretical Aspects of the SOM Algorithm". arXiv:0704.1696v1. 13 Apr 2007

[4] Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." "Representation Learning with Contrastive Predictive Coding" arXiv preprint arXiv:1807.03748, 2018.

[5] Sohn, Kihyuk. "Improved deep metric learning with multi-class n-pair loss objective." NeurIPS, 2016.

[6] Hénaff, Olivier J., Ali Razavi, Carl Doersch, S. M. Eslami, and Aaron van den Oord. "Data-efficient image recognition with contrastive predictive coding." arXiv preprint arXiv:1905.09272, 2019

[7] He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." arXiv preprint arXiv:1911.05722, 2019

[8] Bachman, Philip, R. Devon Hjelm, and William Buchwalter. "Learning representations by maximizing mutual information across views." NeurIPS, 2019.

[9] Tian, Yonglong, Dilip Krishnan, and Phillip Isola. "Contrastive multiviewcoding." arXiv preprint arXiv:1906.05849, 2019

[10] Sermanet, Pierre, Corey Lynch, Jasmine Hsu, and Sergey Levine. "Time-contrastive networks: Self-supervised learning from multi-view observation." CVPRW, 2017.

[11] Poole, Ben, Sherjil Ozair, Aaron van den Oord, Alexander A. Alemi, and George Tucker. "On variational bounds of mutual information." ICML, 2019.

[12] Chen, Ting, Simon Kornblith, Kevin Swersky, et al. "Big Self-Supervised Models Are Strong Semi-Supervised Learners." ArXiv:2006.10029 [Cs, Stat], June 2020. arXiv.org, http://arxiv.org/abs/2006.10029

[13] Grill, Jean-Bastien, et al. "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." ArXiv:2006.07733 [Cs, Stat], June 2020. arXiv.org, http://arxiv.org/abs/2006.07733.

[14] Chen, Xinlei, et al. "Improved Baselines with Momentum Contrastive Learning." ArXiv:2003.04297 [Cs], Mar. 2020. arXiv.org, http://arxiv.org/abs/2003.04297

[15] Caron, Mathilde, et al. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments." ArXiv: 2006.09882.

[16] Jing, Longlong, and Yingli Tian. "Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey." ArXiv:1902.06162 [Cs], Feb. 2019. arXiv.org, http://arxiv.org/abs/1902.06162

[17] Caron, Mathilde, et al. "Deep Clustering for Unsupervised Learning of Visual Features." ArXiv:1807.05520 [Cs], Mar. 2019. arXiv.org, http://arxiv.org/abs/1807.05520

[18] Asano, Yuki Markus, et al. "Self-Labelling via Simultaneous Clustering and Representation Learning." ArXiv:1911.05371 [Cs], Feb. 2020. arXiv.org, http://arxiv.org/abs/1911.05371

[19] Xie, Qizhe, et al. "Unsupervised Data Augmentation for Consistency Training." ArXiv:1904.12848 [Cs, Stat], June 2020. arXiv.org, http://arxiv.org/abs/1904.12848

[20] Zhirong Wu, et al. "Unsupervised Feature Learning via Non-Parametric Instance Discrimination." ArXiv:1805.01978 [cs.CV], May 2018. arXiv.org, https://arxiv.org/abs/1805.01978

[21] Ishan Misra, et al. "Self-Supervised Learning of Pretext-Invariant Representations." ArXiv:1912.01991. 4 Dec 2019.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations.” ArXiv 2002.05709 [cs.CV]. last revised 1 Jul 2020.
[23] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stéphane Deny,” Barlow Twins: Self-Supervised Learning via Redundancy Reduction”, arXiv:2103.03230, last revised 14 Jun 2021.
[24] Xinlei Chen, Kaiming He,” Exploring Simple Siamese Representation Learning”, arXiv:2011.10566, 20 Nov 2020.
[25] Marie Cottrell, Madalina Olteanu, Fabrice Rossi, Nathalie Villa-Vialaneix. Self-Organizing Maps, theory and applications. Revista de Investigacion Operacional, 2018, 39 (1), pp.1-22. ffhal-01796059
[26] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, Nicu Sebe,” Whitening for Self-Supervised Representation Learning”, arXiv:2007.06346, 14 May 2021.
[27] Mirko Klukas, Marcus Lewis and Ila Fiete “Efficient and Flexible Representation of Higher-Dimensional Cognitive Variables with Grid Cells” Published in PLOS Computational Biology Journal . 2020/04/28.
[28] L.J.P. van der Maaten and G.E. Hinton, “Visualizing High-Dimensional Data Using t-SNE”. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.
[29] Jayanth Reddy Regatti, Aniket Anand Deshmukh, Eren Manavoglu, Urun Dogan,"Consensus Clustering with Unsupervised Representation Learning". arXiv:2010.01245. Last revised 8 Jul 2021
[30] Geoffrey Hinton,"How to represent part-whole hierarchies in a neural network". arXiv:2102.12627.25 Feb 2021.
[31] https://discuss.pytorch.org/t/depthwise-convolutions-from-tf-to-pytorch/11682. Visited August 2021.
[32] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, Armand Joulin, "Emerging Properties in Self-Supervised Vision Transformers". arXiv:2104.14294.24 May 2021.
[33] Adrien Bardes, Jean Ponce, Yann LeCun,"VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning", arXiv:2105.04906.11 May 2021.
[34] Robert M. Mok and Bradley C. Love, "A non-spatial account of place and grid cells based on clustering models of concept learning". Nature Communications volume 10, Article number: 5685.12 December 2019.
[35] Florent Forest, Mustapha Lebbah, Hanene Azzag & Jérôme Lacaille, "Deep embedded self-organizing maps for joint representation learning and topology-preserving clustering". Neural Computing and Applications (2021). 03 August 2021