A Survey of Self-Supervised Learning Applied to Visual Assessment of Cluster Tendency

(Project report for E9 246 Advanced Image Processing, Jan-Apr 2023)

Paritosh Tiwari, 20917, PhD, RBCCPS
M Samuel Jayakumar, 21181, M.Tech. A.I.

1 Problem Description

In this project, we focus on the problem of clustering to identify natural groupings of objects in said data. Clustering is a process of grouping similar objects or records in order to capture the underlying patterns if any. It has traditionally focused on finding groups among objects of the same type, such as clusters of retail customers (objects) based on the products they have bought (attributes).

Deep learning based clustering techniques have recently acquired popularity in this domain due to their ability to handle high-dimensional data, which is common in many real-world applications. There are a variety of state-of-the-art deep learning based clustering methods. Some of these methods use neural networks to learn an encoding of the data and then use traditional clustering methods on the encoded data, while others use neural networks to perform clustering directly. One popular approach is deep embedded clustering (DEC) [10] which utilises an autoencoder neural network to learn a low-dimensional representation of the data, and then uses this representation for clustering. Another approach is the deep clustering network (DCN) [12] which directly performs clustering by learning a neural network that maps the input data to a probability distribution over clusters. There are also methods that incorporate generative models like variational autoencoders (VAEs) [4] and generative adversarial networks (GANs) [5] for unsupervised clustering. These methods can learn the underlying distribution of the data and generate new samples that belong to a specific cluster.

However, deep clustering methods have several limitations. The most important one is that they explicitly require the number of clusters to be specified beforehand, which can be difficult to determine in practice, especially when dealing with large and complex datasets without ground-truth information. In this project, we aim to develop a framework to process high-dimensional data into a lower dimension [2]. We will generate embeddings/representations of the data and then proceed to evaluate the clusters visually and not numerically, by generating highly interpretable visual representations.

2 Methodology

Our approach towards experimentation (Figure 1) is to cover as many variations as possible for the following: *Image datasets, Base SimCLR model configurations and VAT techniques.* Therefore, for each dataset, we used every model configuration and subsequently, every VAT technique at our disposal. Table 1 gives an overview of our selected variations. The main goal

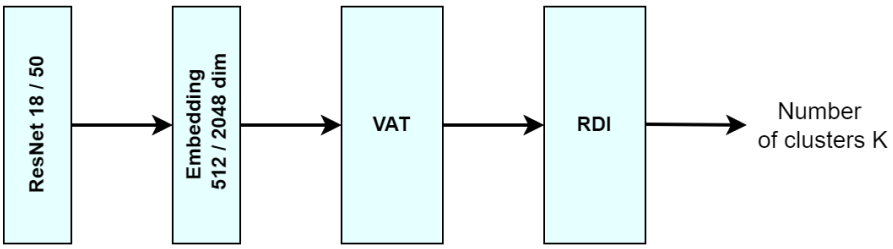


Figure 1: Architecture Workflow

Datasets	MNIST, FashionMNIST, Kuzushiji-MNIST, CIFAR-10
Model Configurations (output dim)	ResNet18 (512), ResNet50 (2048)
VAT Configurations (distance metric)	VAT and iVAT (euclidean), VAT and iVAT (cosine), Spectral VAT

Table 1: Expriment Variations

was to output VAT images with fairly distinguishable dark blocks, which is what the comparison will be based on subsequently. Other possible variations such as loss functions, increased batch sizes and epochs (beyond what we were able to work with) and other encoder models were avoided due to implementation, time and GPU memory constraints.

2.1 Datasets

MNIST: The MNIST database [11] (Modified National Institute of Standards and Technology database) is a large database of handwritten digits. It was created by re-mixing the samples from NIST's original datasets. The database contains 60,000 training images and 10,000 testing images, all grayscaled and of dimensions 28×28.

Fashion MNIST: Fashion-MNIST [9] was intended to serve as a replacement for the original MNIST database for benchmarking machine learning algorithms. The dataset contains 70,000 28×28 grayscale images of fashion products from 10 categories from a dataset of Zalando article images, with 7,000 images per category.

Kuzushiji MNIST: Kuzushiji-MNIST (Japanese letters) [3] is a drop-in replacement for the MNIST dataset (28x28 grayscale, 70,000 images), provided in the original MNIST format as well as a NumPy format. They chose one character to represent each of the 10 rows of Hiragana when creating the dataset.

CIFAR-10: The CIFAR-10 dataset [7] consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

2.2 Contrastive Learning

Contrastive methods aim to learn representations by enforcing similar elements to be equal and dissimilar elements to be different. The core of contrastive learning is the Noise Contrastive

Estimator (NCE) loss.

$$NCE_{Loss} = -log\frac{exp(sim(g(x), g(x^+)))}{exp(sim(g(x), g(x^+))) + \Sigma_{k=1}^{K} exp(sim(g(x), g(x_k^-)))} \quad (1)$$

In the equation above, x^+ as a data point similar to the input x. In other words, the observations x and x^+ are correlated and the pair (x, x^+) represents a positive example. Usually, x^+ is the result of some transformation on x. This can be a geometric transform aimed to change the size, shape or orientation of x, or any type of data augmentation technique. Some examples include *rotation, sheer, resize, cutout and more.* On the other hand, x^- are examples dissimilar to x. The pair (x, x^-) form a negative example and they are meant to be uncorrelated. Here, the NCE loss will enforce them to be different from the positive pairs. Note that for each positive pair (x, x^+) we have a set of K negatives. Indeed, empirical results have shown that a large number of negatives is required to obtain good representations.

The $sim(.)$ function is a similarity (distance) metric. It is responsible for minimizing the difference between the positives while maximizing the difference between positive and negatives. Often, $sim(.)$ is defined in terms of dot products or cosine similarities. Lastly, $g(.)$ is a convolution neural network encoder. It is used to learn embeddings for positive and negative examples. These embeddings are then passed as input to the contrastive loss.

2.3 SimCLR

SimCLR [2] uses the same principles of contrastive learning described in the previous section. It uses contrastive learning to maximize agreement between 2 augmented versions of the same image. Given an input image, we create 2 correlated copies of it, by applying 2 separate data augmentation operators. The transformations include random crop and resize, random colour distortions, and random Gaussian blur. The order of the operations is kept fixed, but since each operation has its own uncertainty, it makes the resulting views visually different. Note that since we apply 2 distinct augmentation functions on the same image, if we sample 5 images, we end up with $2 \times 5 = 10$ augmented observations in the batch.

To maximize the number of negatives, the idea is to pair each image (indexed i) in the batch with all other images (indexed j). Note that we avoid pairing an observation i with itself, and with its augmented version. As a result, for each image in the batch, we get $2 \times (N-1)$ negative pairs where N is the batch size. By arranging negative samples in this way, SimCLR has the advantage of not needing extra logic to mine negatives.

SimCLR uses ResNet-50 as the main ConvNet backbone. The ResNet receives an augmented image of shape $(224, 224, 3)$ and outputs a 2048-dimensional embedding vector h. Then, a projection head $g(.)$ is applied to the embedding vector h which produces a final representation $z = g(h)$. The projection head $g(.)$ is a Multilayer Perceptron (MLP) with 2 dense layers. Both layers have 2048 units and the hidden layer has a non-linearity (ReLU) activation function. For the similarity function, cosine similarity is used which measures the cosine of the angle between 2 non-zero vectors in a d-dimensional space. If the angle between 2 vectors is 0 degrees, the cosine similarity is 1. Otherwise, it outputs a number smaller than 1 all the way down to -1. Note that the contrastive learning loss operates on the latent space mapped by the projection head $g(.)$, the z embedding vectors.

Once the system is trained, we can through away the projection head $g(.)$ and use the representations h (straight from the ResNet) to learn new downstream tasks.

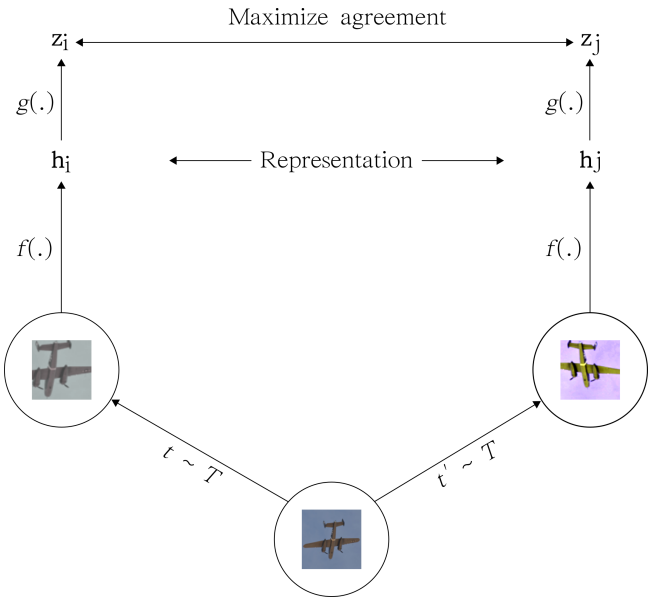


Figure 2: SimCLR framework

2.4 VAT Algorithms

The VAT (Visual Assessment of Cluster Tendency) family of algorithms [1], [6], for visually assessing the cluster tendency of a set of Objects $O = o_1, ..., o_n$ when they are represented either as object vectors or by numerical pairwise dissimilarity values. The objects are reordered and the reordered matrix of pairwise object dissimilarities is displayed as an intensity image. Clusters are indicated by dark blocks of pixels along the diagonal.

Let D be an $n \times n$ dissimilarity matrix corresponding to the set $O = o_1, ..., o_n$. We assume that D satisfies the following (metric) conditions:

$$\forall\ 1 \leq i, j \leq n : D_{ij} \geq 0,\ D_{ij} = D_{ji},\ D_{ii} = 0. \tag{2}$$

We display D as an intensity image I, which we call a dissimilarity image. The intensity or gray level g_{ij} of pixel (i, j) depends on the value of D_{ij}. The value $D_{ij} = 0$ corresponds to $g_{ij} = 0$ (pure black); the value $D_{ij} = D_{max}$, where D_{max} denotes the largest dissimilarity value in D, gives $g_{ij} = D_{max}$ (pure white). Intermediate values of D_{ij} produce pixels with intermediate levels of gray in a set of gray levels $G = G_1, ..., G_m$. The images use 256 equally spaced gray levels, with $G_1 = 0$ (black) and $G_m = D_{max}$ (white). The displayed gray level of pixel (i, j) is the level $g_{ij} \in G$ that is closest to D_{ij}.

The VAT re-ordering algorithm is similar to Prim's algorithm for finding a *minimal spanning tree* (MST) of a weighted graph. Figure 3 shows an example of reordered matrix D_N^* obtained using the VAT algorithm. The generated dark blocks in the image will help us automatically identify clusters.

3 Experiments and Results

Given a dataset, our approach was to train and extract embeddings from the original ResNet architectures and modified ResNets with base model outputs downsampled to 16 units for ResNet18 and 32 units for ResNet50. These extracted embeddings served as inputs to VAT/iVAT

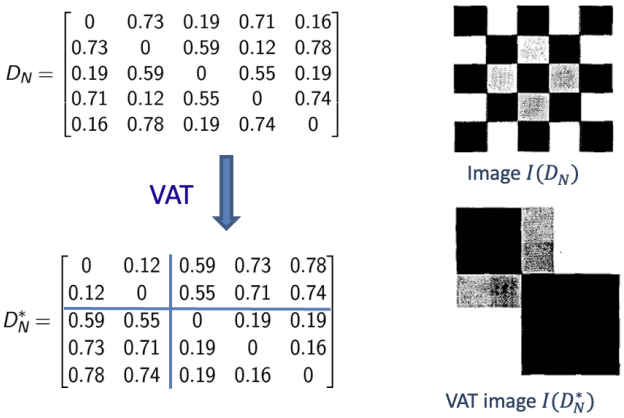


Figure 3: Results of applying the VAT algorithm [1]

algorithms with Euclidean and cosine metrics. We trained all instances with a batch size of 2048 and for 1000 epochs. After training, embeddings were generated for 8000 images and were fed as input to respective VAT algorithms. Tables 2, 3, 4 and 5 show the results of our experiments.

4 Conclusion and Future Work

Overall, the approach taken by us shows a lot of promise, even though the majority of results were unsatisfactory. It is important to note that all of the VAT algorithms when supplied with the raw images datasets as input, generate a plain grey image, not showing any cluster structure, whatsoever. Even though the VAT images in our results do not show the expected number of clusters, we were still able to extract some degree of inherent cluster structure in the image datasets (corresponding to their classes). The results for the CIFAR-10 dataset of coloured RGB images are expectedly worse than the grayscaled datasets, as the colour channels add much more complexity to the embeddings. However, as the work on grayscaled images progresses, so too will the results for coloured images.

Out of all the image datasets, MNIST shows the best results. And out of all the VAT algorithm variants, Spectral VAT [8] shows the most discernable dark blocks across all of the datasets. We will refrain from making a comparison between ResNet18 and ResNet50 as the models showed similar top 1 accuracy during training, ranging from 69 - 74 %.

Proceeding further, we think trying other types of encoder networks like Siamese networks and other loss functions will benefit anyone willing to work in this domain. It is to be stated that the VAT family of algorithms do not handle high dimensional input very well and thus they play a part in the results we have presented. Exploring other VAT variants without improving the separation of datapoints in the embedding layer and the projection head will not yield satisfactory results in our opinion. Furthermore, figuring out a way to somehow downsample the output embeddings from the ResNets will serve towards improving the VAT images. We tried a couple of straightforward downsampling techniques but the results did not improve and thus we chose to omit them.

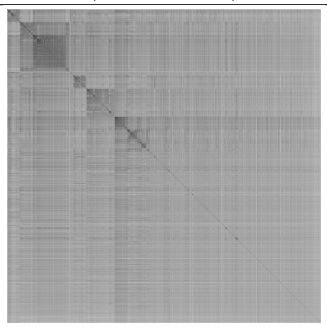
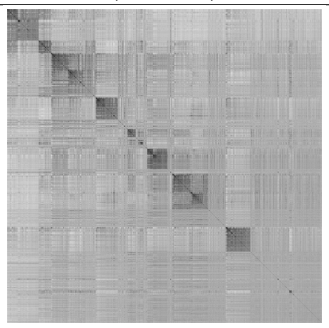
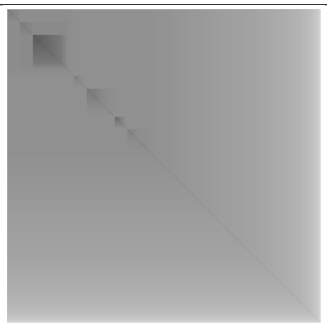
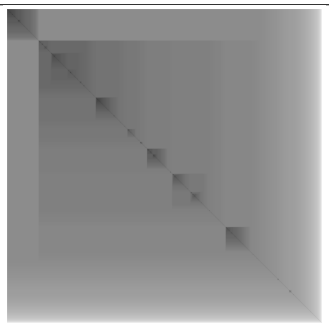
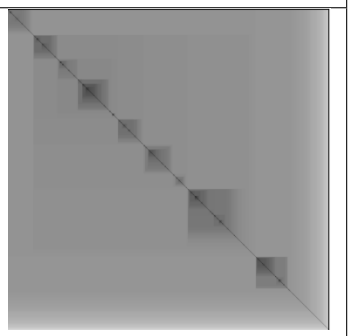
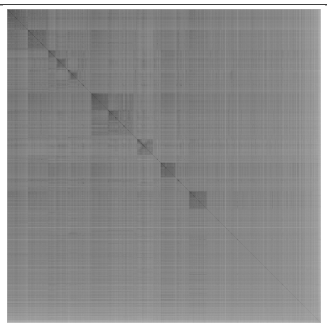
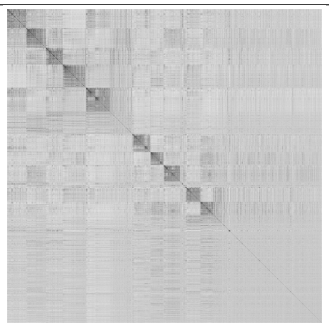
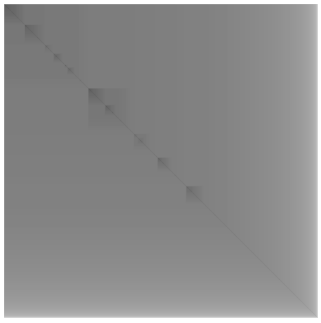
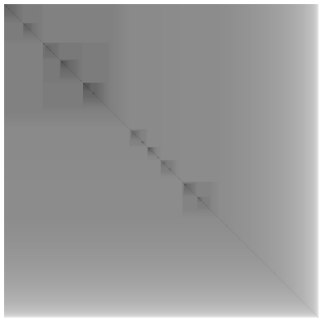
Model	VAT Algorithm	Output Image (Euclidean)	Output Image (Cosine)
ResNet18 512 dim output	VAT		
ResNet18 512 dim output	iVAT		
ResNet18 512 dim output	Spectral VAT		
ResNet50 2048 dim output	VAT		
ResNet50 2048 dim output	iVAT		

Table 2: MNIST experiment results

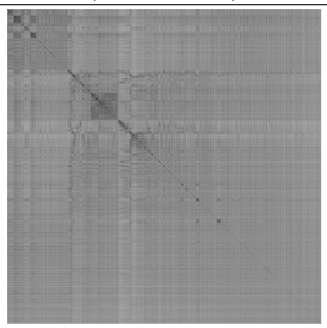
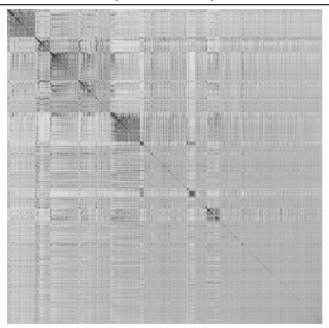
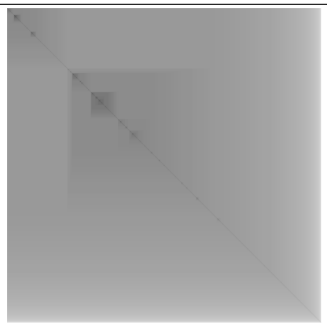
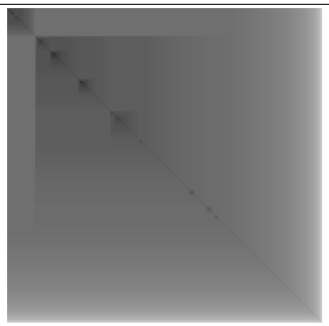
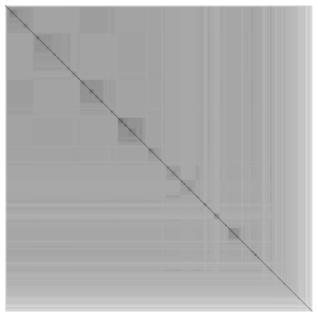
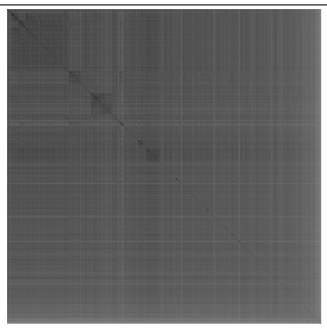
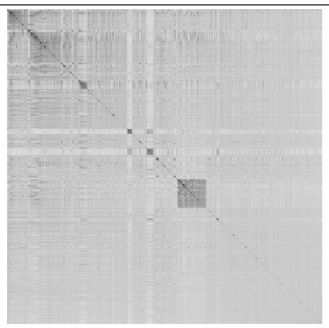
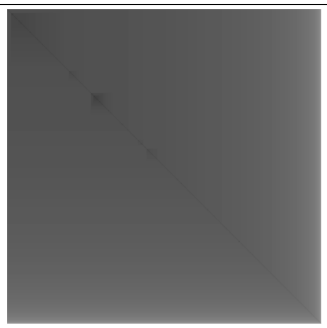
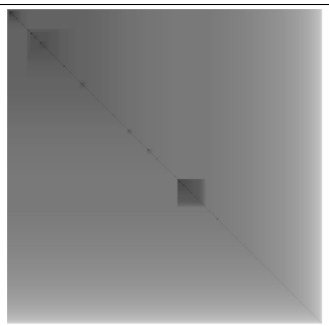
Model	VAT Algorithm	Output Image (Euclidean)	Output Image (Cosine)
ResNet18 512 dim output	VAT		
ResNet18 512 dim output	iVAT		
ResNet18 512 dim output	Spectral VAT		
ResNet50 2048 dim output	VAT		
ResNet50 2048 dim output	iVAT		

Table 3: Fashion MNIST experiment results

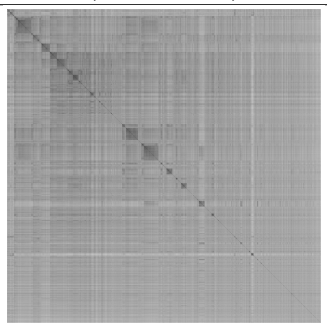
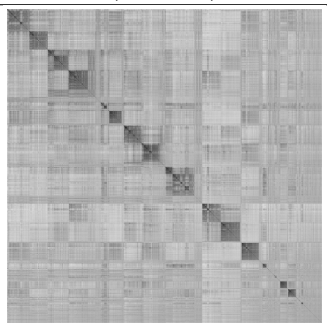
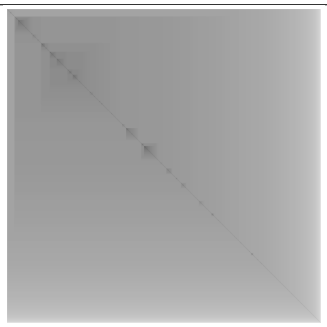
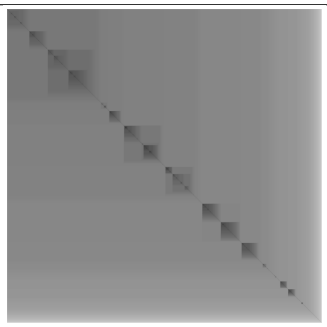
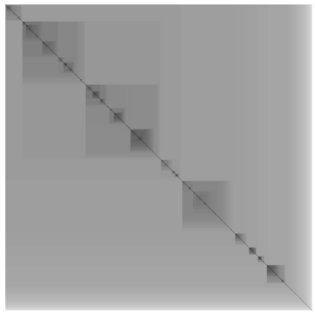
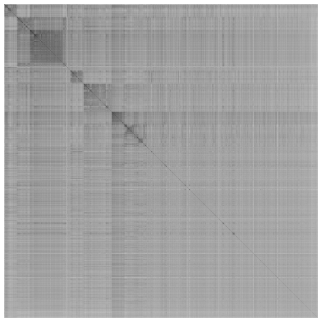
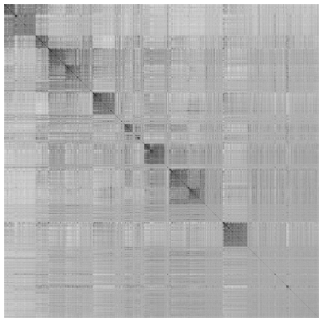
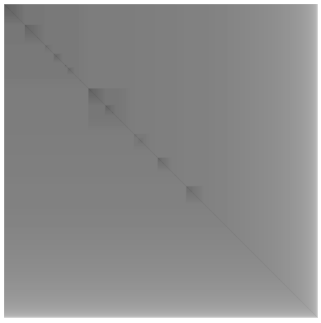
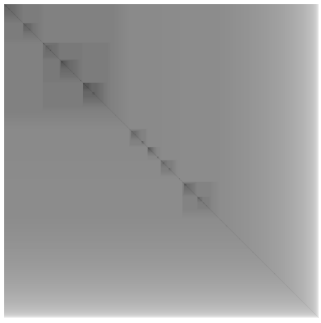
Model	VAT Algorithm	Output Image (Euclidean)	Output Image (Cosine)
ResNet18 512 dim output	VAT		
ResNet18 512 dim output	iVAT		
ResNet18 512 dim output	Spectral VAT		
ResNet50 2048 dim output	VAT		
ResNet50 2048 dim output	iVAT		

Table 4: Kuzushiji MNIST experiment results

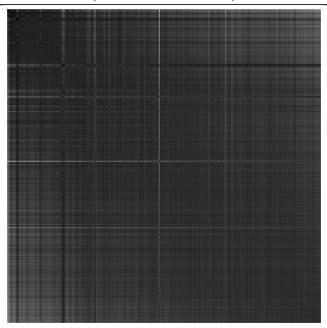
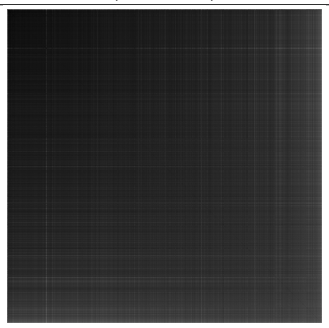
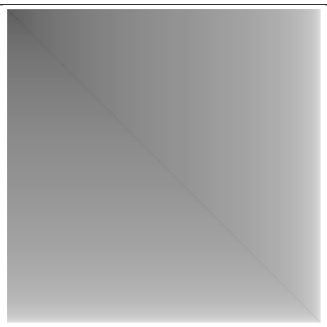
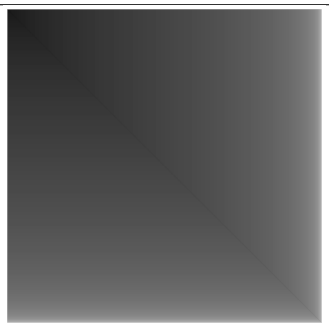
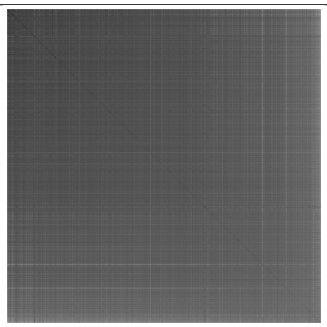
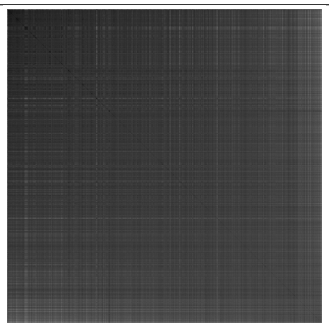
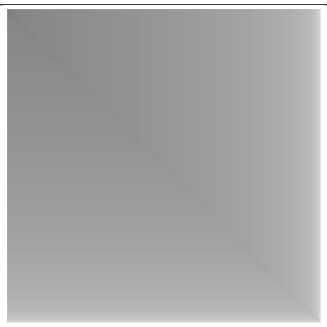
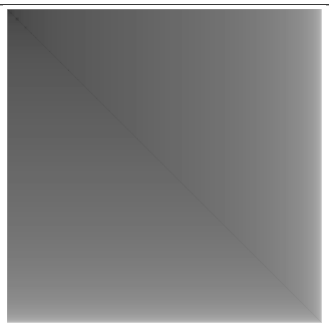
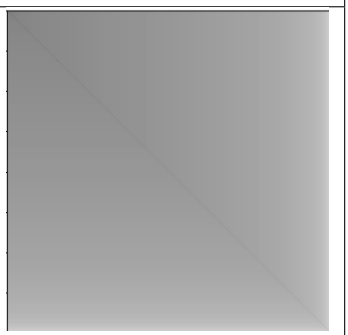
Model	VAT Algorithm	Output Image (Euclidean)	Output Image (Cosine)
ResNet50 2048 dim output	VAT		
ResNet50 2048 dim output	iVAT		
ResNet50 32 dim output	VAT		
ResNet50 32 dim output	iVAT		
ResNet50 32 dim output	Spectral VAT		

Table 5: CIFAR-10 experiment results

References

[1] Bezdek, J., Hathaway, R.: Vat: a tool for visual assessment of (cluster) tendency. In: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (2002)

[2] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations (2020)

[3] Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., Ha, D.: Deep learning for classical japanese literature (2018)

[4] Diederik P Kingma, M.W.: Auto-encoding variational bayes. arxiv. 2013.

[5] Goodfellow, e.a.: Generative adversarial networks. arxiv. 2014.

[6] Havens, T.C., Bezdek, J.C.: An efficient formulation of the improved visual assessment of cluster tendency (ivat) algorithm. IEEE Transactions on Knowledge and Data Engineering (2012)

[7] Krizhevsky, A.: Cifar-10: Learning multiple layers of features from tiny images (2009)

[8] Wang, L., Geng, X., Bezdek, J., Leckie, C., Kotagiri, R.: Specvat: Enhanced visual cluster analysis. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 638–647 (2008)

[9] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)

[10] Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. arxiv.

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

[12] Yang, e.a.: Towards k-means-friendly spaces: Simultaneous deep learning and clustering. icml-2017.