

Evaluating the impact of data representation on t-SNE projections

Liviu-Ştefan Neacşu-Miclea

*Department of Computer Science, Babeş-Bolyai University
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania
E-mail: nlic3194@scs.ubbcluj.ro*

Abstract – *Data visualization in computer vision is helpful in uncovering concealed patterns in the dataset distribution. Since a set of images exists in a large dimensional space, the research in this domain greatly benefits from dimensionality reduction techniques. This study focuses on using t-SNE as an image feature visualization tool. We use three labeled datasets (Fashion MNIST, Muffin vs Chihuahua, Large-Scale Fish Dataset) to examine whether a t-SNE projection could extract relevant information from a handful of encodings: an autoencoder latent space, a pretrained VGG16, and the features processed by a CNN classifier. We use stress metrics and Shepard diagrams to analyze and compare the results. We conclude that t-SNE projections capture key global structures that can assist in CV and classification tasks.*

1. Introduction

Dimensionality reduction is a key step in understanding complicated relationships in higher dimensional datasets [SMK24]. Finding a good bi-dimensional mapping of multidimensional data has practical applications in a variety of domains. Multidimensional Scaling (MDS), Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), t-distributed Stochastic Neighbor Embedding t-SNE are examples of some popular data projection techniques [SMK24] [Esp+21]. The current work focuses on t-SNE and attempts to test its ability of creating meaningful plots on datasets with different quirks (large size – Fashion MNIST, raw uncropped unsegmented real world images – Muffin vs Chihuahua, visually and structurally similar images taken in a controlled environment where the important features are localized texture and color information – Large Scale Fish Dataset). For each dataset, a couple of different encodings of the same data were chosen to create intermediary representation spaces before passing them to the t-SNE projection algorithm. First, the latent space of an identity auto-encoder is used as a simple means of compression since working with high resolution pixel images would be computationally expensive. Then, the feature maps embeddings of a pretrained VGG-16 are used as reference data, due to the proven abilities of the CNN network in feature extraction. Finally, the embeddings of a per-dataset custom trained classifier are created, which are expected to be the most well organized representation of the dataset. Of course, projecting into lower dimensional space comes with a loss of information [Gho+20]. For this reason, stress metrics (NS, SNS, NMS) and Shepard goodness score (SGS) are used to evaluate the fidelity of the projection reproducing the global structure in the real data. This study has educational purpose as being the first contact of the author with t-SNE and dimensionality reduction, however, it leads to some interesting conclusions, such as providing a way to assess the complexity and ease of classification in a dataset, identifying overfitting and differences between subsets of the samples.

2. Related work

2.1. t-SNE

t-distributed Stochastic Neighbor Embedding (t-SNE) is a frequently used [SMK24] nonlinear dimensionality reduction statistical method designed which enables projecting high dimensional data into a lower dimensional space [Gho+20]. It applies a probability-based method to represent the similarities among data points in both high- and low-dimensional spaces by taking into account nearby points and disregarding more distant points [SMK24]. The pairwise distances between data

points are modeled into a Student-t distribution with one degree of freedom (which solves the “crowding problem” of its previous SNE methods [Gho+20]), thus the similarity between two points can be expressed as the probability of them being neighbors, considering neighbors are following this distribution. In a formal manner, for each pair of data point x_i and x_j is assigned a similarity probability

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

which tells the probability that x_j is a neighbor of x_i where σ_i denotes the variance of a Gaussian distribution P_i centered in x_i , a value which is determined by matching the distribution’s perplexity $Perp(P_i) = \prod_j p_{j|i}^{-p_{j|i}}$ with the one provided as an user defined parameter, while $p_{i|i} = 0$ [Gho+20]. Then, $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$ is defined. The t-SNE algorithm tries to find the points y_i in a lower dimension whose similarities

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}},$$

involving pairwise probabilities following a Student-t distribution with one degree of freedom, try to approximate p_{ij} [Gho+20]. The points y_i are obtained by minimizing the Kullback-Leibler divergence applied on distributions P and Q [Gho+20].

2.2. Auto-encoder

An auto-encoder (AE) is a neural network consisting of an encoder E that maps inputs to a latent space of lower dimension (also called bottleneck), and a decoder D which uses the latent space to construct an output [Zha18]. Depending on the function they serve, multiple classes of auto-encoders emerge, some memorable mentions being the denoising (DAE) and variational auto-encoders (VAE). A convolutional auto-encoder describes D and E as convolutional networks (CNNs) [Zha18]. The AE plays a great role in data compression, for example, if one trains the AE on the identity function, $D(E(x)) \approx x$, then the latent space acts as a low dimensional embedding (compression) of the original input.

2.3. Projection metrics

Projecting data to a lower dimensional space almost certainly leads to a deformed representation which may not capture all the information existent in the original data (such as distances or clusters) [SMK24]. Quality metrics were developed to help assess the relevance of a dimensionality reduction process by evaluating how well a certain aspect is captured by the projection [SMK24].

Raw stress (RS), like the other stress metrics, is a global metric which measures the difference between pairwise distances in the high and low dimensional spaces:

$$RS(X, P) = \sum_{i,j} \left(\Delta^X(x_i, x_j) - \Delta^P(p_i, p_j) \right)^2,$$

where X, P are the sets of original and projected data, and Δ^X, Δ^P denote a distance function (this work will only use the Euclidean norm) [SMK24]. It is easily observed that RS is scale dependent (changing the scale of the projection space will result in a different metric value, or formally, if for $\alpha \in \mathbb{R}$ we

define $\alpha P = \{\alpha p_i | p_i \in P\}$, then $RS(X, \alpha P)$ depends on α , for which other stress variants were introduced to mitigate this problem. It is not used as a metric in the current work, however it is mentioned in order to provide an intuition of the how the next described metrics work.

Normalized stress (NS) tackles the problem of RS producing high stress values even for acceptable projections. Although originally designed as an optimization function, it has been used in literature as a metric due to its lower output magnitude [SMK24]. The normalization occurs as a scaling of the raw stress according to the sum of distances in the original space:

$$NS(X, P) = \sqrt{\frac{\sum_{i,j} (\Delta^X(x_i, x_j) - \Delta^P(p_i, p_j))^2}{\sum_{i,j} \Delta^X(x_i, x_j)^2}} \quad [\text{SMK24}].$$

However, NS still remains scale sensitive, therefore **Scale-normalized Stress (SNS)** was invented, which minimizes the value of normalized stress over all possible scales: $SNS(X, P) = \min_{\alpha > 0} NS(X, \alpha P)$ [SMK24].

Shepard Goodness Score (SGS) consists of a 2D diagram plot of distances in the two spaces $(\Delta^X(x_i, x_j), \Delta^P(p_i, p_j))_{1 \leq i < j \leq N}$ and its associated Spearman rank correlation. The visual interpretation of the diagram is that the distance preservation is better as the plotted point cloud is closer to the first diagonal [SMK24]. The Spearman rank correlation coefficient, which describes how well the relations between distances can be approximated by a monotonic function, is a sub-unitary measure, making it independent of the data scale [SMK24].

Non-metric Stress (NMS), also known as Kruskal stress, measures the way order is preserved after the projection:

$$NMS(X, P) = \frac{\sum_{i,j} (\Delta^{\hat{X}}(\hat{x}_i, \hat{x}_j) - \Delta^P(p_i, p_j))^2}{\sum_{i,j} \Delta^P(p_i, p_j)^2},$$

where $\Delta^{\hat{X}}(\hat{x}_i, \hat{x}_j) = f(\Delta^X(x_i, x_j))$, where f is obtained by applying isotonic regression on the Shepard diagram [SMK24]. It was noted that NMS is scale-invariant [SMK24].

3. Proposed method

3.1. Used datasets

In order to accurately demonstrate the utility of t-SNE plots, three image classification datasets from unrelated domains were chosen such that they pose challenges of different nature and complexity. A bar plot of per-label samples quantity for each dataset can be consulted in *Fig. 1*.

Fashion MNIST (FMNIST) is a collection of 60,000/10,000 train/test 28x28px grayscale images, structurally similar with the original handwritten digits MNIST dataset, but featuring 10 classes of clothes and fashion accessories [XRV17]. Just like MNIST, Fashion MNIST was designed for benchmarking machine learning algorithms [XRV17]. In the current work, FMNIST is used as a reference dataset, due to its curated and balanced nature. It is expected that the proposed method would lead to best results on this dataset.

Muffin vs. Chihuahua (MvsC) is an iconic binary classification problem, also extensively known for the related internet memes and its usage as a beginner's tutorial dataset. The problem is built around visual similarities between Chihuahuas and blueberry muffins. The used dataset [Cor]

contains 4733/1184 train/test images belonging to these 2 classes. The samples are raw unsegmented images of diverse sizes and orientations obtained from a search engine [Cor]. The data diversity is a strong point of considering the choice of the dataset.

A Large Scale Fish Dataset (LSFD) is an initiative of researchers from Izmir University of Economics, Izmir, Turkey, to create a practical seafood classification dataset featuring their local specialities [UUT20]. For each of the 9 classes of fish species, there are 1000 pre-augmented (rotated/flipped) size-normalized train images and 50 raw high-resolution test samples (exception Trout class, with 30 test images) [UUT20]. Each sample was captures in the same real-world environment setting (a single fish centered on a blue reflective background material). As opposed to the other two datasets, this one excels in visual uniformity and would make it easier even for traditional non-ML approaches to extract features needed in classification.

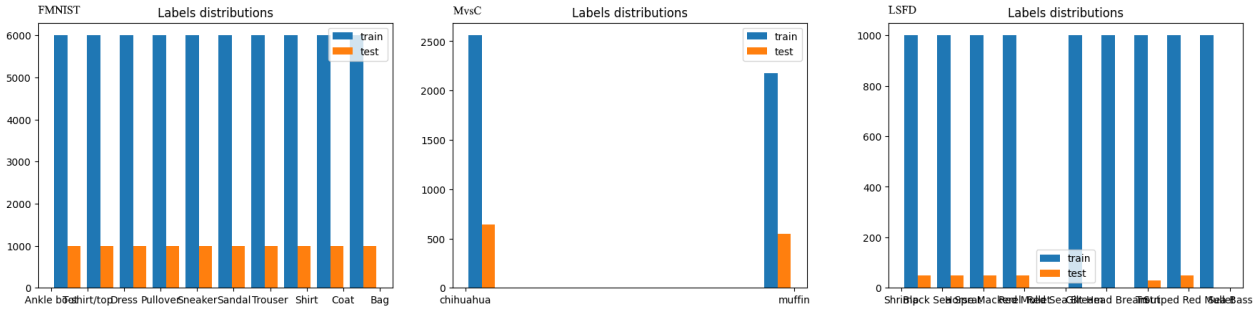


Fig.1. Labels distribution for each dataset

3.2. Experiment workflow

The proposed goal is to see how t-SNE behaves under different representations of the same data. Figure 2 supplements the method description with a graphical visualization. First, one would like to directly project the images onto a t-SNE plot, however, since the data dimensionality would be big enough to challenge technical capabilities, an intermediate unsupervised dimensionality reduction

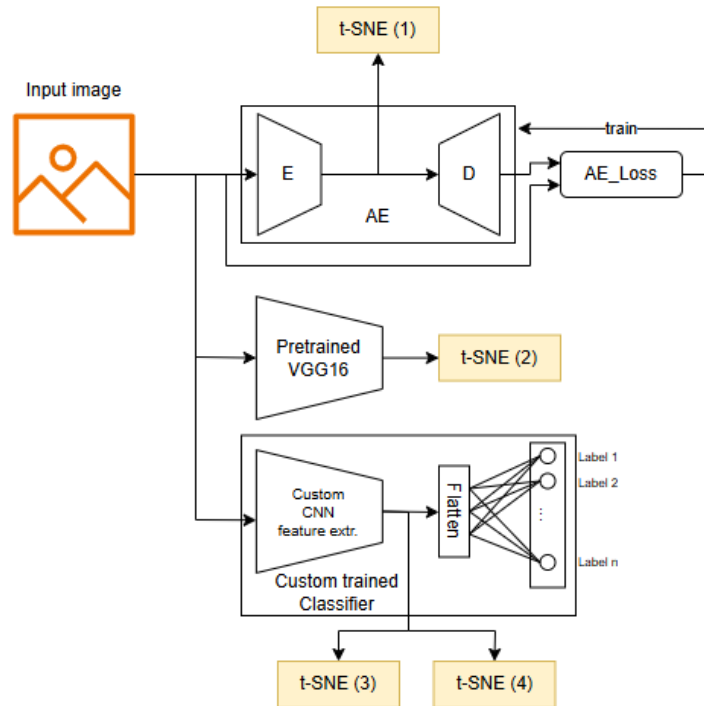


Fig.2. Workflow diagram of the experiment

method was chosen instead. Note that while this may be considered unnecessary for FashionMNIST’s small resolutions, this step was kept anyway for consistency with the other two datasets and a more correct comparison. Therefore, for each labeled dataset (X, Y) , an auto-encoder $AE = (E, D)$ is trained to fit the identity function of the target image set (X, X) and a first projection t-SNE (1) is built from the embeddings $E(X)$. Then, a VGG16 model pretrained on the ImageNET dataset [SZ15], known for its extraordinary feature extraction abilities, is used as a direct state-of-the-art reference to compare the AE embeddings against. The second plot t-SNE (2) is obtained from the output of the VGG network, where the last 3 fully-connected layers were scrapped. Finally, a single custom LeNet-inspired network with softmax activation was trained for the classification task, and the activations of the last feature map are plotted onto t-SNE (3) (train data) and t-SNE (4) (validation data). More information about architectures and training of the mentioned models are given in the next section.

Every t-SNE projection is supplied with the evaluated metrics described in *Section 2.3*, a Shepard diagram and an interpretation taking account visual details and numerical values. All t-SNE reduction algorithms use a manually established perplexity of 100 which was found as a sweet spot between plot expressiveness and execution time. Due to system performance limitations, up to 10^4 samples were randomly selected for computing the metrics, since obtaining the pairwise distances requires $\Theta(N^2)$ computation time and memory. This change only affects the FMNIST dataset, however, we argue that a uniform random sample from such a balanced dataset would also be balanced on its own, therefore measuring it instead of the original dataset should not alter the general perspective. Also, due to the same reasons, out of the at most $10^4 \times 10^4$ pairwise distances, only at most 5000 were selected to be scattered on the Shepard diagrams. All the experiments have been performed on the Kaggle online platform with GPU P100 accelerator, 30GB RAM and the execution time is in the hours’ landmark.

3.3. Training details

Dataset	Original image size	Rescaled input size	Latent sp. dim.
FMNIST	$28 \times 28 \times 1$	$32 \times 32 \times 3$	128
MvsC	3 channels, unconstrained size	$160 \times 160 \times 3$	2048
LSFD	$590 \times 445 \times 3$	$192 \times 192 \times 3$	2048

Table 1. Data shapes used during training

Dataset	Epochs	LR (Adam opt)	Content Loss
FMNIST	3*	10^{-4}	perceptual, 1 layer
MvsC	10	10^{-3}	perceptual, 7 layers
LSFD	10	10^{-3}	perceptual, 2 layers

Table 2. Autoencoder training metaparameters

* The number of epochs was adjusted to the dataset size.

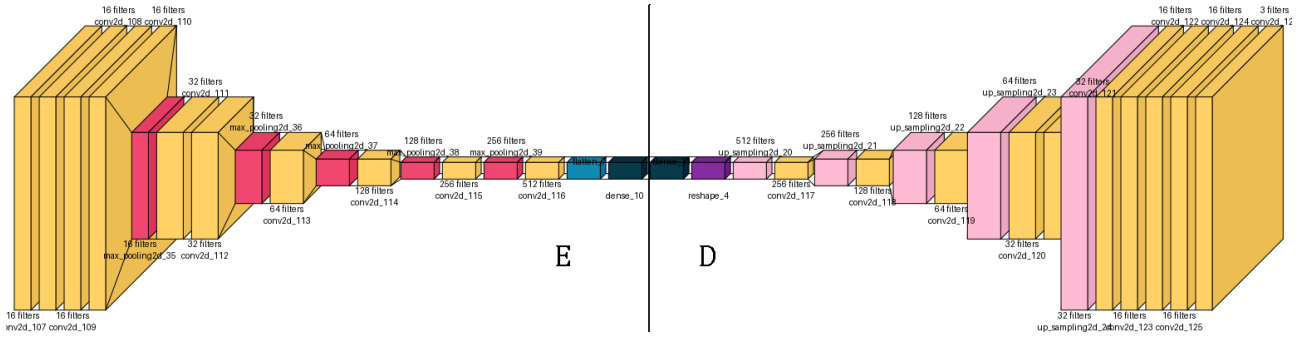


Fig. 3. Auto-encoder architecture

The auto-encoder (Fig. 3) follows a symmetrical structure. All the images have been normalized so that pixel values are brought in the positive sub-unitary interval. All convolutions are zero-padded (such that the spatial dimensions do not change) with 3×3 kernel and LeakyReLU activation (except for the decoder’s last convolution, which is activated by $hardsigmoid(x) = clip_{[0,1]}(\frac{x}{6} + \frac{1}{2})$). Much interest is accorded in capturing better shape information, thus there are more low-filter stacked convolutions at the top of the model than deeper high-filter ones. The down sampling is done using 2×2 MaxPooling2D layers, while the process is reversed through nearest-neighbor UpSampling2D layers. There are no fixed input shapes across the three datasets, however, in order to keep shapes consistent with the presence of the 5 down/up sampling layers, the width and height of the images was imposed to be a multiple of 32. In addition, the latent space (embeddings) dimensions was empirically chosen to be 2048, except for FMNIST, where a lower dimensional space was selected because of small input size. *Table 1* provides details about training time shapes of data. *Table 2* summarizes the training process of the AE. The auto-encoder loss from *Figure 1* is defined as $AE_Loss(X_{real}, X_{pred}) = ContentLoss_L(X_{real}, X_{pred}) + w \cdot SSIM(X_{real}, X_{pred})$, where $ContentLoss_L(X, Z) = MSE(VGG_{1..L}(X), VGG_{1..L}(Z))$ is the perceptual loss, inspired from style transfer literature [HB17] (mean-squared error between the feature maps seen through the first L layers of VGG16 computed for all pixels and channels across the instance), $SSIM(X, Z)$ is the image similarity index and w is a scaling hyperparameter (defaulted to 10).

The classifier (*Fig. 4*) performs feature extractions through 7 convolutional layers with the same setup as above and 4 max-pooling layers. A final Dense layer with soft-max following the flatten feature activations predicts the probability of each class being present in the input. The model was naturally

trained with categorical cross-entropy loss and Adam optimizer with learning rate 10^{-3} . Tables N1 and N2 provide a detailed layer summary about the used architectures of the two convolutional networks.

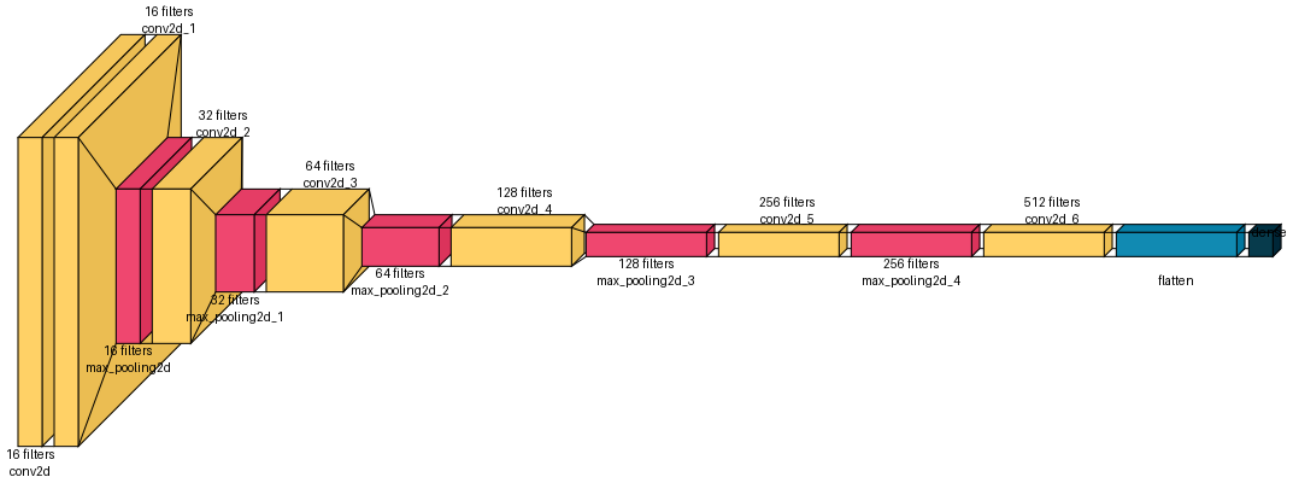


Fig.4. CNN Classifier architecture

Input, 32Nx32M pixels 3 channels
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 32 filters, 3x3 kernel
Conv2D, 32 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 64 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 128 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 512 filters, 3x3 kernel
Flatten
Dense, latent space, tanh activ.
Dense, leaky_relu activ. (reverse flatten)
UpSampling2D, 2x2
Conv2D, 256 filters, 3x3 kernel
UpSampling2D, 2x2
Conv2D, 128 filters, 3x3 kernel
UpSampling2D, 2x2
Conv2D, 64 filters, 3x3 kernel
UpSampling2D, 2x2
Conv2D, 32 filters, 3x3 kernel
Conv2D, 32 filters, 3x3 kernel
UpSampling2D, 2x2
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
Conv2D, 3 filters, 1x1 kernel, hard_sigmoid

Table N1. Autoencoder summary

Input, 32Nx32M pixels 3 channels
Conv2D, 16 filters, 3x3 kernel
Conv2D, 16 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 32 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 64 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 128 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 256 filters, 3x3 kernel
MaxPooling2D, 2x2
Conv2D, 512 filters, 3x3 kernel
Flatten
Dense, number of classes neuron, softmax

Table N2.CNN Classifier summary

4.1. Models performance

Figure 5 showcases the auto-encoder performance, while the classifier details can be consulted in Figure 4, along with the classification metrics exposed in Table 3. It can be observed that the AE can

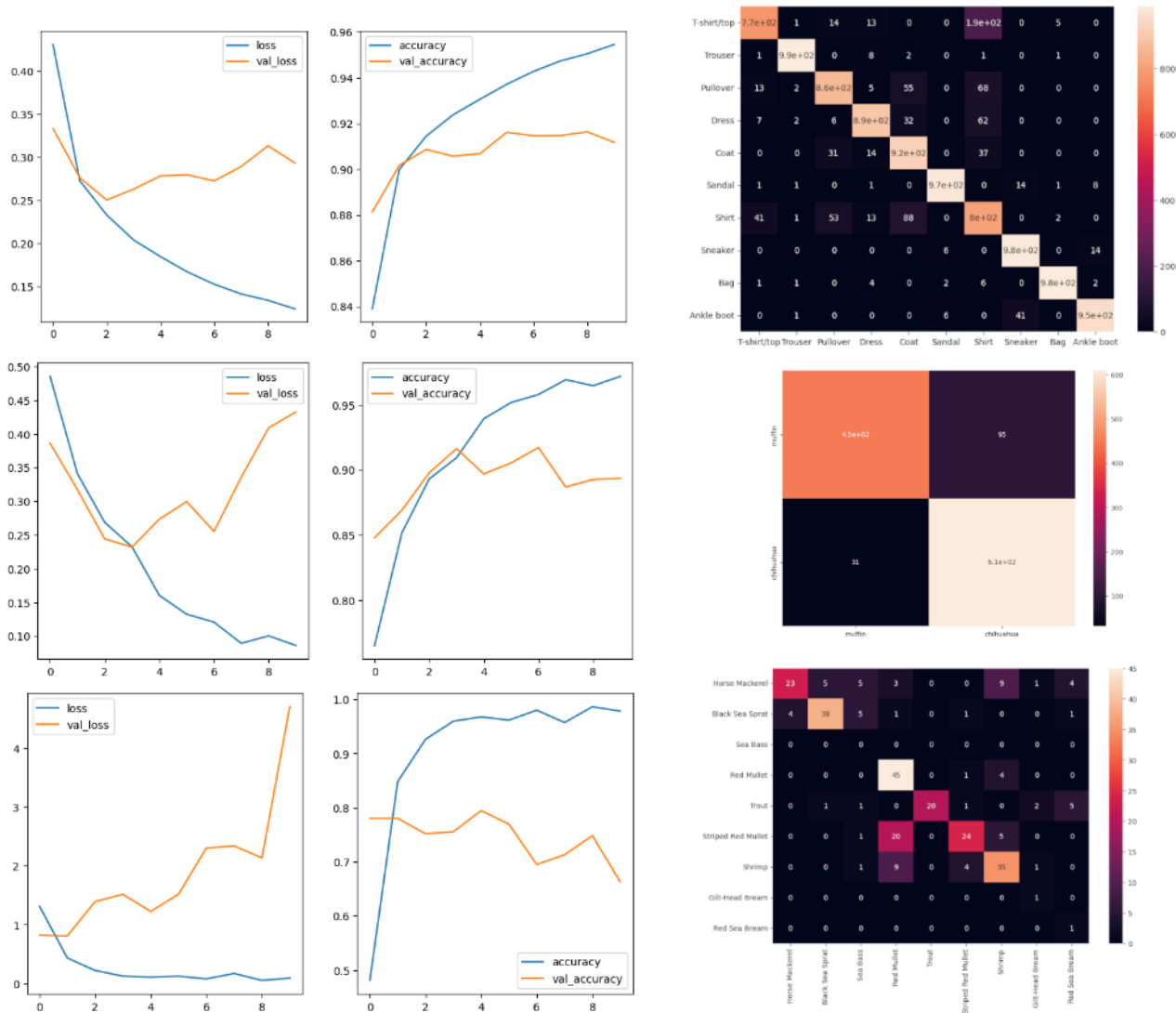


Fig.4. Classifier training results

Row 1: FMNIST, Row 2: CvsM, Row3: LSFD

Col 1: loss, Col 2: accuracy, Col 3: Confusion matrix (test data)

learn the general shape of the object. MvsC dataset has the poorest reconstructions, due to the higher variety of contents distribution, as opposed to the other two datasets, where the overall geometric structure is more predictable. While in FMNIST the blurriness of the output makes the object in the reconstructed image still recognizable, the process becomes harder in LSFD, where only shape and an average color on the fish’s skin are preserved, and next to impossible in MvsC.

<i>FMNIST</i>										
	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Boot
Precision	0.7754	0.9870	0.8574	0.8908	0.9181	0.9738	0.8016	0.9800	0.9839	0.9519
Recall	0.9232	0.9909	0.8921	0.9388	0.8386	0.9857	0.6872	0.9468	0.9908	0.9753
F1	0.8429	0.9890	0.8744	0.9142	0.8766	0.9797	0.7400	0.9631	0.9874	0.9634
IoU	0.7284	0.9782	0.7768	0.8420	0.7803	0.9603	0.5873	0.9289	0.9751	0.9295
Accuracy	0.9120									

MvsC									
	Muffin				Chihuahua				
Precision	0.8256				0.9516				
Recall	0.9355				0.8652				
F1	0.8771				0.9063				
IoU	0.7812				0.8288				
Accuracy	0.8937								
LSFD									
	H. Mckrl	B.S.Sprat	Sea Bass	Red Mullet	Trout	S.R. Mullet	Shrimp	G-H. Bream	R.S. Bream
Precision	0.4600	0.7600	0	0.9000	0.6666	0.4800	0.7000	1.0000	1.0000
Recall	0.8518	0.8636	0	0.5769	1.0000	0.7741	0.6603	0.2000	0.0909
F1	0.5974	0.8085	0	0.7031	0.8000	0.5925	0.6796	0.3333	0.1666
IoU	0.4259	0.6785	0	0.5421	0.6666	0.4210	0.5147	0.2000	0.0909
Accuracy	0.6631								

Table 3. Classification results for each dataset

The FMNIST classifier offers high performance results, with very few drawbacks (the imprecision of distinguishing between Shift and T-shirt). On a similar note, the MvsC binary classification succeeds

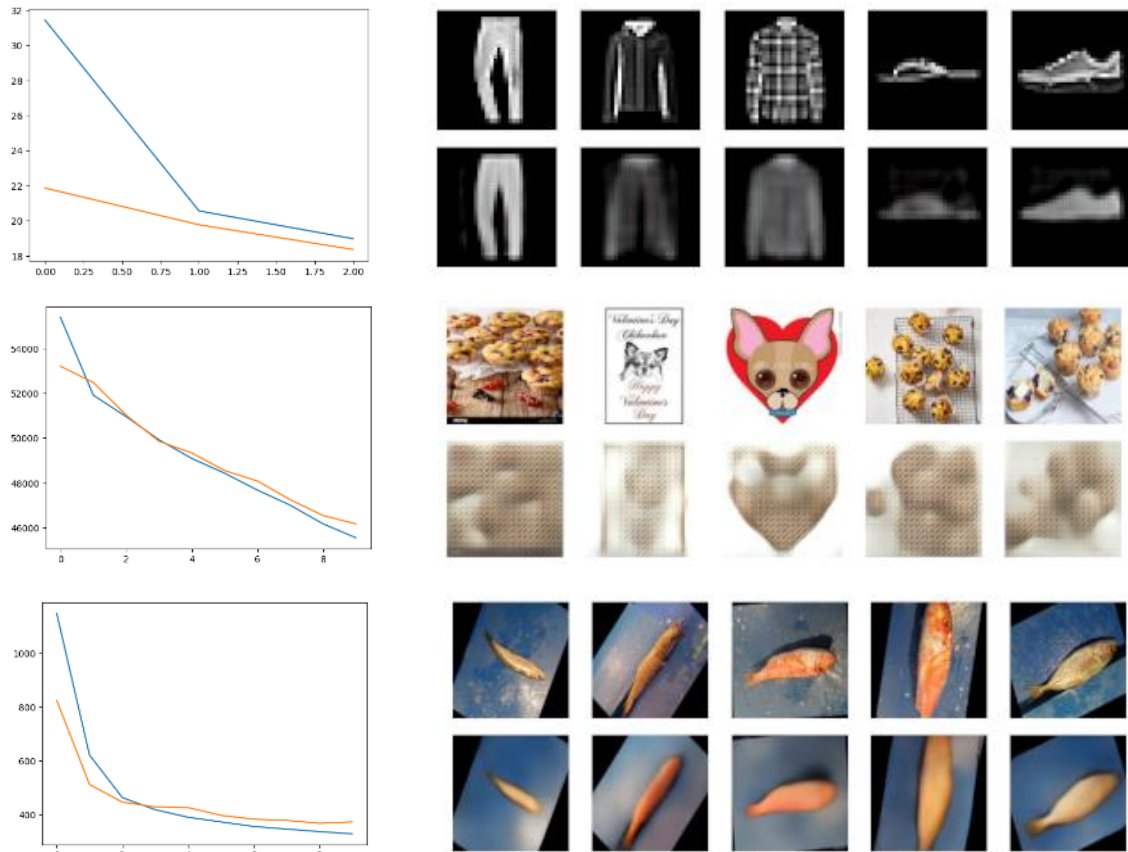


Fig.5. Autoencoder training results

Row 1: FMNIST, Row 2: CvsM, Row3: LSFD

Plot lines (AE loss): Blue = train, Orange = test

with an accuracy of nearly 90%. However, LSFD proves its difficulty by leading to bigger overfitting and very unreliable classification. Part of the explanation is the fact that the train and validation sets are not processed in the same way (while the train set has been pre-augmented by the creators [UUT20], the validation set was adapted as part of this experiment in order to match the same rotate and scale transforms). Moreover, some classes have very few validation samples, while for Sea Bass none are provided. It can be checked that the same classifier trained on 90%/10% train-test split only on the augmented data would have a performance comparable with the other datasets.

4.2. t-SNE plots

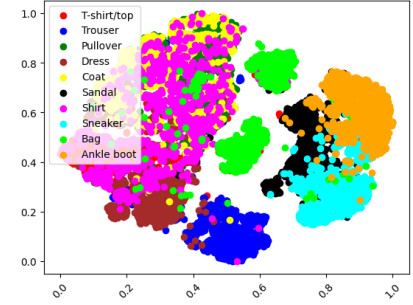
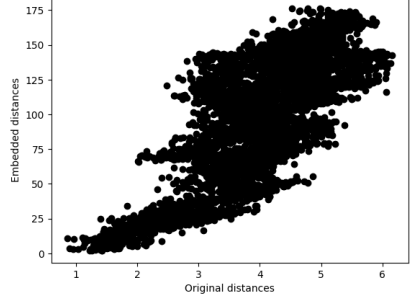
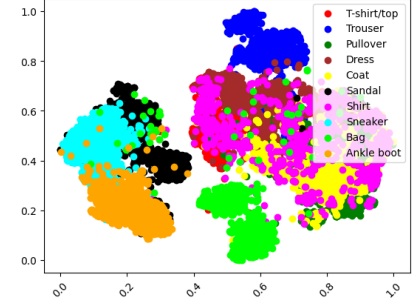
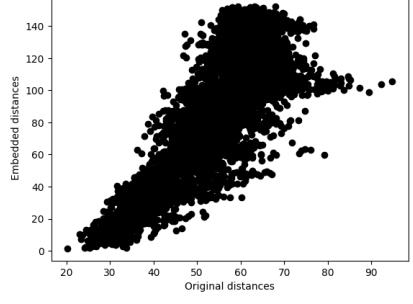
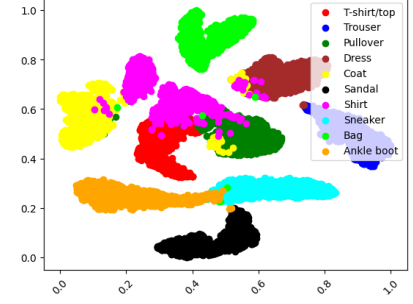
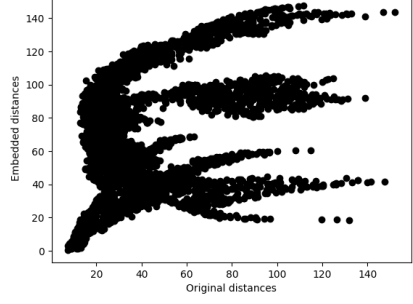
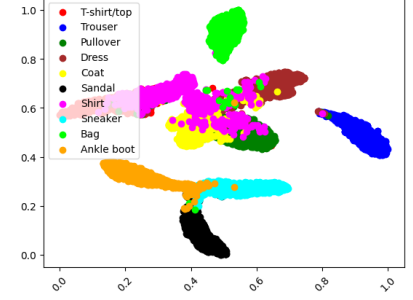
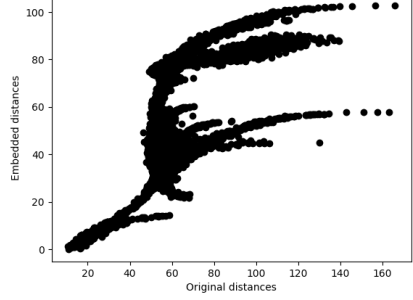
Step	t-SNE plot	Shepard diagram	Metrics
AE embeddings (t-SNE (1))			NS = 22.7105 SGS = 0.6678 SNS = 0.3114 NMS = 0.3077
VGG-16 embeddings (t-SNE (2))			NS = 0.5895 SGS = 0.44816 SNS = 0.3838 NMS = 0.3692
Classifier embeddings – train (t-SNE (3))			NS = 0.5829 SGS = 0.5244 SNS = 0.4246 NMS = 0.3563
Classifier embeddings – test (t-SNE (4))			NS = 0.3884 SGS = 0.6331 SNS = 0.3879 NMS = 0.3456

Table 4. t-SNE plots and measurements for FMNIST

Tables 4, 5, and 6 showcase the t-SNE projection, the Shepard diagram and the projection metrics on each of the (1) ... (4) capture phases. A visual inspection of the plots reveals that the AE and VGG embeddings lead to noisy cluster structures, while the classifier’s feature maps project to more organized, almost clearly separable aggregations. In all phases, the t-SNE projection on the FMNIST dataset clearly identifies classes of cumulated fashion items (chest-covering clothes, shoes, trousers and bags), which explains the lower classifier precision rates of the labels who belong to the same mega-cluster (e.g. shirt vs t-shirt vs pullover vs dress).

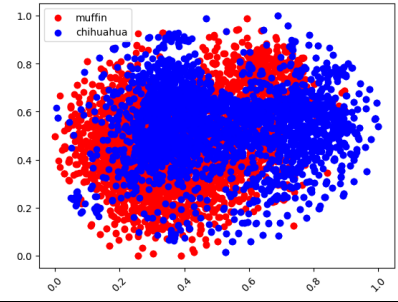
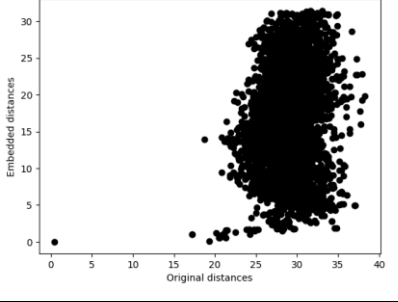
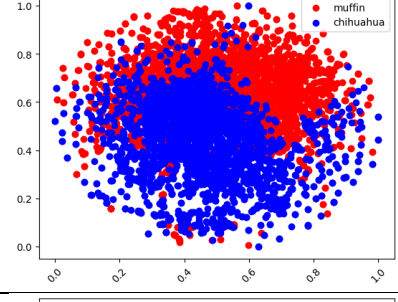
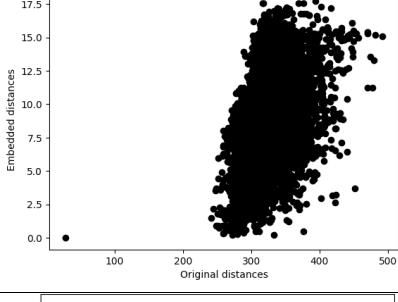
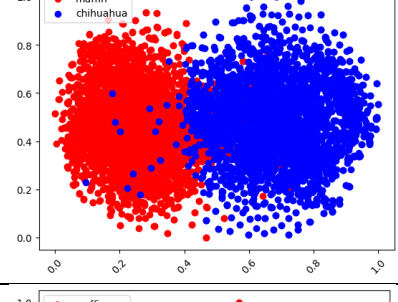
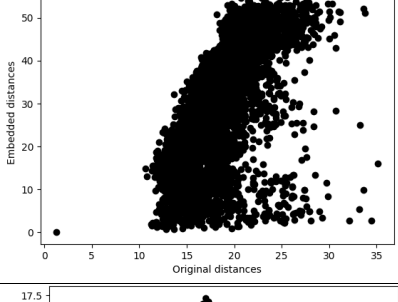
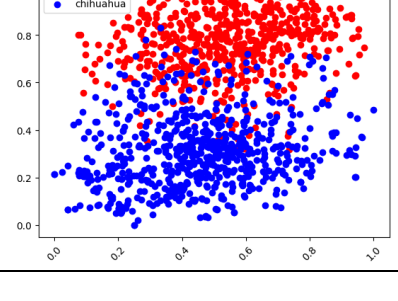
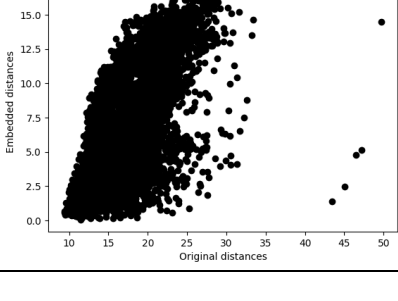
Step	t-SNE plot	Shepard diagram	Metrics
AE embeddings (t-SNE (1))			NS = 0.5822 SGS = 0.5081 SNS = 0.4063 NMS = 0.3927
VGG-16 embeddings (t-SNE (2))			NS = 0.9778 SGS = 0.3268 SNS = 0.42915 NMS = 0.4255
Classifier embeddings – train (t-SNE (3))			NS = 0.5582 SGS = 0.5613 SNS = 0.4190 NMS = 0.4106
Classifier embeddings – test (t-SNE (4))			NS = 0.6172 SGS = 0.6077 SNS = 0.4031 NMS = 0.3875

Table 5. t-SNE plots and measurements for *MvsC*

In the case of *MvsC* dataset (Table 5), the AE and VGG embeddings are not able to clearly separate the two classes (although VGG produces a slightly more coagulated map). However, the classifier embeddings project to clean clusters in the train variation, while this may not be clear on the validation set unless a clustering method involving points density is employed.

An interesting phenomenon occurs in LSFD, where both AE and VGG pay attention to the rotation angles used in image augmentation, thus the symmetric and radial projections in the first two rows of Table 6. Despite some local groupings of same-class instances being present, they have no impact to the global structure of the projection (supposing labels are not provided, one could not tell anything about the number of classes and their distribution). When plotting the classifier embeddings, the same-class points seem to bring themselves into less fragmented regions, however, the radial structure still persists (although not as symmetrical as before) and there is still a lack of geometric frontier between clusters. The acute dissimilarity between train and test classification embeddings

complements the fact that these two sets do not model the same distribution, therefore the previously mentioned classifier’s drop in accuracy.

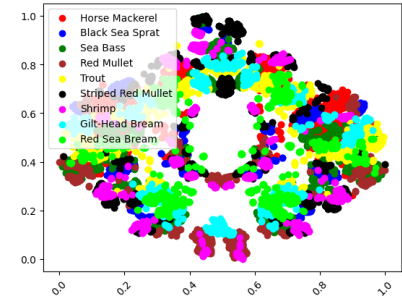
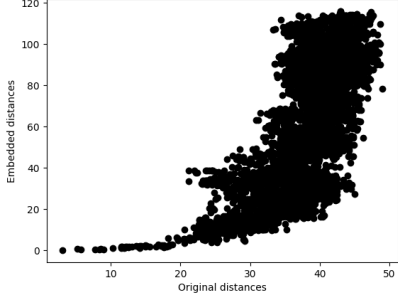
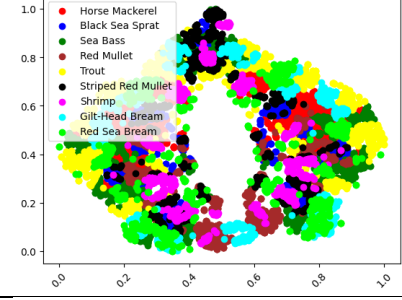
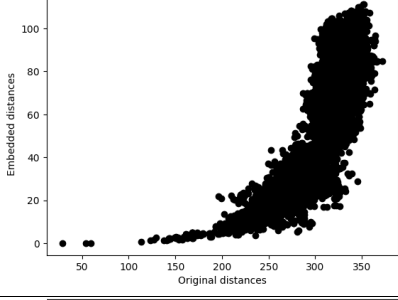
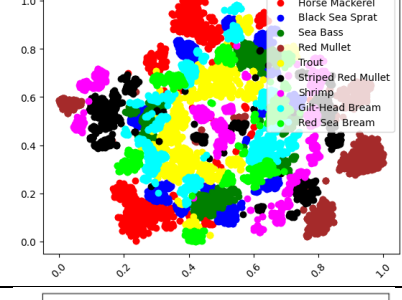
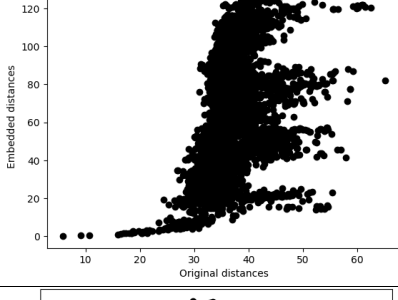
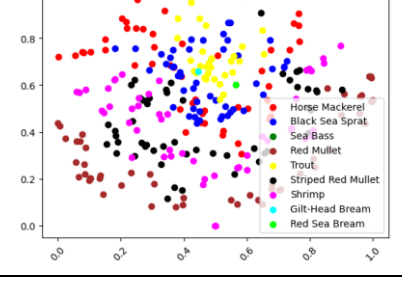
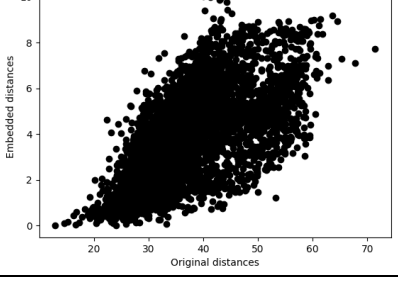
Step	t-SNE plot	Shepard diagram	Metrics
AE embeddings (t-SNE (1))			NS = 0.8448 SGS = 0.5431 SNS = 0.3684 NMS = 0.3338
VGG-16 embeddings (t-SNE (2))			NS = 0.7918 SGS = 0.6626 SNS = 0.3717 NMS = 0.3011
Classifier embeddings – train (t-SNE (3))			NS = 1.0561 SGS = 0.4176 SNS = 0.3873 NMS = 0.3841
Classifier embeddings – test (t-SNE (4))			NS = 0.8909 SGS = 0.6682 SNS = 0.3518 NMS = 0.3287

Table 6. t-SNE plots and measurements for LSFD

The Shepard diagram reveals additional information about the dataset in cause. For example, it helps in identifying possibly duplicate samples in the train MvsC dataset (both real and projected distances between some pair of points is 0 while the rest of the data follows a completely different structure), despite the author claiming that they removed identical samples [Cor]. The same can be stated about LSFD, however with a greater degree of transition continuity, which might be a natural consequence of the performed image augmentation affine transforms. All Shepard diagrams (with the exception of MvsC, AE and VGG) show a monotonic tendency which indicates preservation of distances after the projection, relative to the global picture. An intriguing pattern occurs in the case of t-SNE (3) for all dataset, where the Shepard diagram seems to be a union of independent curvy-stripe-shaped clouds of points. The behavior is clearly observed in projections FMNIST (3) and (4), LSFD (3) and faintly in MvsC (3) and (4). The heads of some horizontal stripes can also be distinguished in FMNIST (1)

and LSFD (1). Moreover, each of these stripes seems to converge to some upper limit and their number based on density regions of convergence is close to the number of classes in the dataset. Despite the lower value of the goodness score, such behavior might be a sign of a good classifier. *Figure 7* illustrates the dependence found between SGS and accuracy.

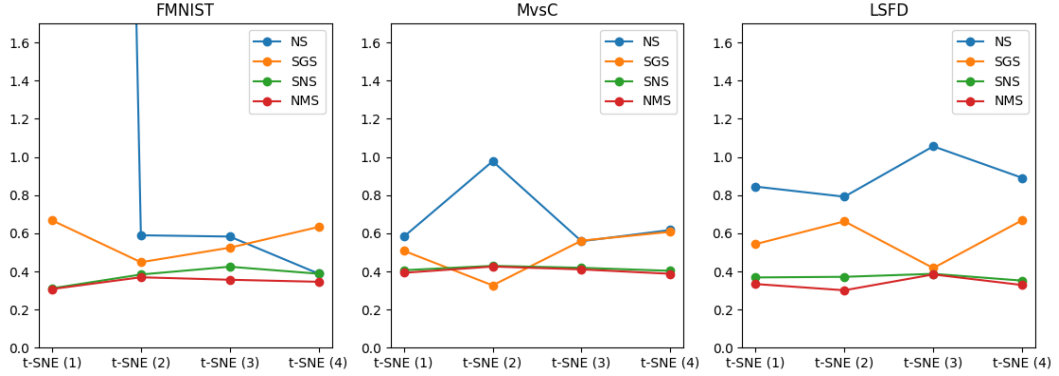


Figure 6. Projection metrics evolution over the four phases

Figure 6 exposes the relationship between the four projection metrics for each dataset in each of the four phases. The notable correlation between Non-Metric Stress and Scale-Normalized stress is also confirmed by the SNS authors [SMK24]. The SGS values indicate a bad to acceptable correlation. The impact of the representation type on the stress metrics depends on the flavors of the dataset, but some recurring patterns can be extracted from the three experiment, such as the normalized stress increases with the number of samples, or that t-SNE (4) has slightly lower average stress than t-SNE (3).

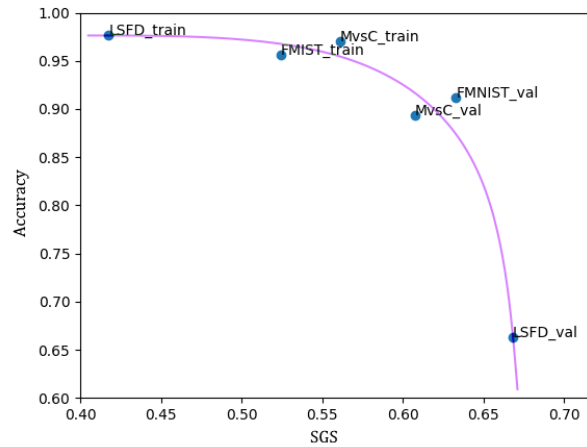


Figure 7. Relation between SGS and Accuracy

5. Conclusion and future work

It has been remarked [SMK24] that interpreting t-SNE plots as-is may lead to untruthful observations, because they may tend to output groups points even where there is no expected cluster structure. However, t-SNE proved its effectiveness in extracting precious information from the chosen datasets. While some of them are useful in shaping a general idea about the data (FMNIST (1)) or evaluating the sanity of the classification (LSFD (3), (4)), failures occur when unprocessed diversity is enhanced (MvsC (1), (2)) or other structural features which should typically be learned to be ignored in classification are given a higher importance (LSFD (1), (2)). Therefore, a reduced visualization of feature spaces are useful both in the data analysis step, where an unsupervised method could obtain

important results about the samples distribution, and in the method evaluation step, when the classes for each sample are predicted, and one could combine them with the projection, on whose basis may decide to open the ways for further investigation.

There are obviously possibilities to improve the current work. The auto-encoder could be better optimized, especially on MvsC. Creating more clean versions of some datasets, with no duplicates, a proper segmentation and consistency in train/validation setups, as well as designing and training a better CNN network, would definitely improve the classification accuracy. Also, more datasets could be analyzed to confirm the occurrence and interpretation of the currently observed patterns and possibly find other concealed correlations. Furthermore, involving local and inter-cluster metrics would come in benefit of having a complete picture over the analyzed data. Moreover, it would be a natural speculation to ask whether the same behavior can be observed in other projection methods like UMAP, MDS or PCA. Last but not least, the current method is computationally expensive with long execution time and excessive memory usage, thus an alternate way to reveal a similar information digest would be welcome.

6. References

- [Cor] S. Cortinhas. Muffin vs Chihuahua image classification dataset. <https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification>. Accessed: 2024-11-17.
- [Esp+21] M. Espadoto et al. “Toward a Quantitative Survey of Dimension Reduction Techniques”. In: IEEE Transactions on Visualization and Computer Graphics 27.3 (2021), pp. 2153–2173. doi: 10.1109/TVCG.2019.2944182.
- [Gho+20] B. Ghogh et al. “Stochastic neighbor embedding with Gaussian and student-t distributions: Tutorial and survey”. In: arXiv preprint arXiv:2009.10301 (2020).
- [HB17] X. Huang and S. Belongie. “Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization”. In: 2017 IEEE International Conference on Computer Vision (ICCV). 2017, pp. 1510–1519. doi: 10.1109/ICCV.2017.167.
- [PSL20] G. G. Pihlgren, F. Sandin, and M. Liwicki. “Improving Image Auto-encoder Embeddings with Perceptual Loss”. In: 2020 International Joint Conference on Neural Networks (IJCNN). 2020, pp. 1–7. doi: 10.1109/IJCNN48605.2020.9207431.
- [SMK24] K. J. Smelser, J. Miller, and S. G. Kobourov. ““Normalized Stress” is Not Normalized: How to Interpret Stress Correctly”. In: ArXiv abs/2408.07724 (2024). url: <https://api.semanticscholar.org/CorpusID:271874691>.
- [SZ15] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: Computational and Biological Learning Society, 2015, pp. 1–14.
- [UT20] O. Ulucan, D. Ulucan, and M. Turkan. “A Large-Scale Dataset for Fish Segmentation and Classification”. In: Oct. 2020. doi: 10.1109/ASYU50717.2020.9259867.
- [XRV17] H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: arXiv preprint arXiv:1708.07747 (2017).
- [Zha18] Y. Zhang. “A better auto-encoder for image: Convolutional auto-encoder”. In: ICONIP17-DCEC. Available online: http://users.cecs.anu.edu.au/Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper58.pdf (accessed on 23 March 2017). 2018.

Acknowledgement:

This work is the result of my own activity, and I confirm I have neither given, nor received unauthorized assistance for this work.

I declare that I did not use generative AI or automated tools in the creation of content or drafting of this document.