NEURAL CLUSTERING BY PREDICTING AND COPYING NOISE

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a neural clustering model that jointly learns both latent features and how they cluster. Unlike similar methods our model does not require a predefined number of clusters. Using a supervised approach, we agglomerate latent features towards randomly sampled targets within the same space whilst progressively removing the targets until we are left with only targets which represent cluster centroids. To show the behavior of our model across different modalities we apply our model on both text and image data and achieve state-of-the-art results on MNIST. Finally, we also provide results against baseline models for fashion-MNIST, the 20 newsgroups dataset, and a Twitter dataset we ourselves create.[1]

1 INTRODUCTION

Clustering is one of the fundamental problems of unsupervised learning. It involves the grouping of items into clusters such that items within the same cluster are more similar than items in different clusters. Crucially, the ability to do this often hinges upon learning latent features in the input data which can be used to differentiate items from each other in some feature space. Two key questions thus arise: *How do we decide upon cluster membership?* and *How do we learn good representations of data in feature space?*

Spurred initially by studies into the division of animals into taxa (Sokol & Sneath, 1963), cluster analysis matured as a field in the subsequent decades with the advent of various models. These included distribution-based models, such as Gaussian mixture models (Duda et al., 1973); density-based models, such as DBSCAN (Ester et al., 1996); centroid-based models, such as k-means.[2] and hierarchical models, including agglomerative (Orloci, 1967) and divisive models (Gower, 1967).

While the cluster analysis community has focused on the unsupervised learning of *cluster membership*, the deep learning community has a long history of unsupervised *representation learning*, yielding models such as variational autoencoders (Kingma & Welling, 2013), generative adversarial networks (Goodfellow et al., 2014), and vector space word models (Mikolov et al., 2013).

In this paper, we propose using noise as targets for agglomerative clustering (or NATAC). As in Bojanowski & Joulin (2017) we begin by sampling points in features space called *noise targets* which we match with latent features. During training we progressively remove targets and thus agglomerate latent features around fewer and fewer target centroids. To tackle the instability of such training we augment our objective with an auxiliary loss which prevents the model from collapsing and helps it learn better representations. We explore the performance of our model across different modalities in Section 3.

Recently, there have been several attempts at jointly learning both cluster membership and good representations using end-to-end differentiable methods. Similarly to us, Yang et al. (2016) use a policy to move points around in feature space but they use a different agglomerative strategy. Liao et al. (2016) propose jointly learning representations and clusters by using a k-means style objective. Xie et al. (2016) introduce deep embedding clustering (DEC) which learns a mapping from data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective

[1] IDs of tweets we used can be found in: https://github.com/neuralclusteringNAT/paper-resources/tree/master/tweet_clustering

[2] A good historical review of k-means can be found in Hans-Hermann (2008).

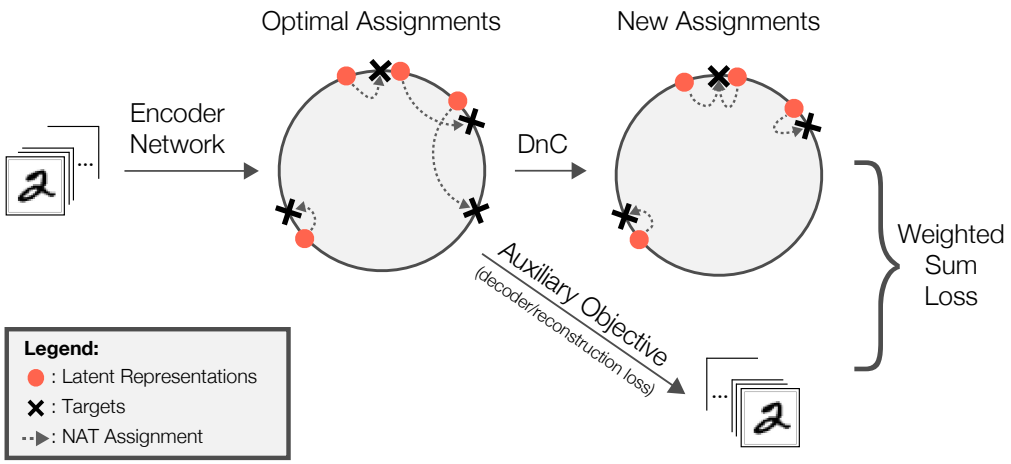

Figure 1: Training diagram for a NATAC model. In this case the model is clustering MNIST digits and uses an autoencoder loss as the auxiliary objective. The delete-and-copy policy removes the target on the bottom-right of the sphere and copies the target at the top. This leads to an agglomeration of the two latent representations at the top of the sphere into the same cluster.

(however, as opposed to the hard assignment we use, they optimize based on soft assignment). Unlike us, all of the above papers require a predefined number of clusters.

2 Model

We begin by discussing the use of noise as targets (NAT) introduced in Bojanowski & Joulin (2017) which is crucial to the understanding of our model. We then describe the intuition behind our approach, then proceed to describe the mechanism itself.

2.1 Noise as Targets (NAT)

Bojanowski & Joulin (2017) proposed a new form of unsupervised learning re-posited as a supervised problem where the task is to learn an encoder function $f_\theta : X \rightarrow Z$ from input space to a latent representation space. The objective is to minimize the L_2 loss between representations $z_i \in Z$, where points in Z are unit normalized, and corresponding targets $y_k \in Y$ where the targets are uniformly sampled from the L_2 unit sphere (thus inhabiting the same space as the z_i). Instead of being tied to corresponding representations as in classic regression, targets are one-to-one re-assigned to different representations after every training step so as to minimize the total sum of distances between matched pairs (this is done using the Hungarian method (Kuhn, 1955)). This pushes the representations z_i of similar inputs $x_i \in X$ to neighborhoods of similar targets y_k.

The motivation behind using NAT was to learn unsupervised features from input data. They show their method performs on par with state-of-the-art unsupervised representation learning methods.

2.2 Noise as Targets for Agglomerative Clustering (NATAC)

Viewed from a clustering perspective we can think of the targets y_i as cluster centroids to which the latent representations z_i are (one-to-one) assigned. Note that although the method introduced in Bojanowski & Joulin (2017) brings the representations of similar x_i closer together (by matching and moving them closer to neighborhoods of similar targets) it cannot produce many-to-one matchings or match multiple similar z_i with a single centroid thus forming a cluster. Simply changing the re-assignment policy to allow for many-to-one matchings is difficult because it causes the model to collapse in on a single target. In this paper we use the above to propose a new form of neural clus-

tering where we progressively delete targets over time and re-assign representations to other nearby targets, all while keeping the model stable using an auxiliary objective.

Delete-and-copy To be able to cluster latent representations we need a way of assigning them to cluster centroids. We propose doing this via an additional delete-and-copy step which allows for many-to-one matchings. Similarly to the NAT method, we first assign representations $\mathbf{z}$ to targets $\mathbf{y}$ using the Hungarian method. We then remove the assigned target and reassign a *copy* of the next nearest target for each representations with some probability α as in Algorithm 1 (see also Figure 1). This has the effect of not only reassigning targets so as to minimize the distance between matched pairs, but also to encourage the model to allow similar examples to be assigned to the same target. The new assignments are denoted as y_i^{new}. The loss is then defined as:

$$\mathcal{L}_{\text{NAT}}(\theta|x;y) = \sum_i \|y_i^{\text{new}} - z_i\|_2 \tag{1}$$

Auxiliary objective To prevent the model from collapsing to a single point, we introduce an auxiliary objective in addition to the loss between representations and targets. In our case we set the auxiliary objective $\mathcal{L}_{\text{aux}}$ to be the reconstruction loss $\sum_i \|x_i - f_{\text{dec}}(z_i)\|_2$ where f_{dec} is some decoder network. The final objective $\mathcal{L}$ is then a weighted sum of the NAT loss $\mathcal{L}_{\text{NAT}}$ (which in our case is the L_2 loss) and the auxiliary objective $\mathcal{L}_{\text{aux}}$:

$$\mathcal{L}(\theta|x;y) = \lambda \mathcal{L}_{\text{NAT}}(\theta|x;y) + \mathcal{L}_{\text{aux}}(\theta|x) \tag{2}$$

Importantly, the auxiliary objective not only prevents the model from collapsing, it also informs how the model clusters. For example, when clustering images the reconstruction loss encourages the model to cluster on similar pixel values. Alternative forms of auxiliary objectives could allow for a discriminator loss or a classification loss for tackling semi-supervised settings. As our goal is unsupervised clustering, we only consider the reconstruction loss.

Model definition During initialization each example x_i in the dataset is paired with a random target y_i, uniformly sampled from a d-dimensional sphere. The NATAC model is then trained using mini-batches from the dataset. Each training step can be broken down as follows:

1. **Forward step:** The examples $\mathbf{x}$ from a random batch of example-target pairs $\big((x_1, y_1) \dots (x_i, y_i) \dots (x_N, y_N)\big)$ are embedded using an encoder function $f_\theta : \mathbb{R}^n \to \mathbb{R}^d$ into corresponding latent representations $z_i \in \mathbb{R}^d$ where $\|z_i\| = 1$ for all i.
2. **Re-assignment step:** Using the Hungarian method from Kuhn (1955), the representations $\mathbf{z}$ are optimally one-to-one matched with targets $\mathbf{y}$ so as to minimize the total sum of distances between matched pairs in the batch. The newly assigned example-target pairing $\big((x_1, y_1^{\text{opt}}) \dots (x_i, y_i^{\text{opt}}) \dots (x_N, y_N^{\text{opt}})\big)$ has the permutation of labels within the batch to minimize the batch-wise loss.
3. **Delete-and-copy (DnC):** With a probability α, delete the optimal target y_i^{opt} and instead assign the nearest target to z_i (which may be the same target) for each example-target pair in the batch producing $\big((x_1, y_1^{\text{new}}) \dots (x_i, y_i^{\text{new}}) \dots (x_N, y_N^{\text{new}})\big)$ as described in Algorithm 1 (see also Figure 1).
4. **Train and update step:** The L_2 loss between targets and latent representations is taken and combined with the auxiliary loss. Gradients w.r.t. θ are then taken and back-propagated along. Notice that although the (re)-assignment step follows a non-differentiable policy, the model is still end-to-end differentiable. Finally, the new example-target assignments are kept after the training step, and persist into the next training step where they are reassigned again.

2.3 Implementation Details

Stopping criterion During training the number of unique targets is tracked. We stop training when the number of unique targets stops decreasing after an epoch of training.

Algorithm 1: The delete-and-copy policy used in our experiments.

Input : A batch of latent representations $\mathbf{z}$, the optimal NAT assignment $\mathbf{y}^{\text{opt}}$, and a probability of copying α
Output: A new NAT assignment $\mathbf{y}^{\text{new}} = \left(y_1^{\text{new}} \dots y_i^{\text{new}} \dots y_N^{\text{new}}\right)$
for z_i *in* $\boldsymbol{z}$ **do**
  p := sample from $U(0, 1)$
  y^{nearest} := nearest target to z_i
  if $p < \alpha$ **then**
    (y_i^{opt} ***is then deleted, and*** $y^{nearest}$ ***is copied and assigned to*** z_i)
    $y_i^{\text{new}} := y_i^{\text{nearest}}$
  else
    $y_i^{\text{new}} := y_i^{\text{opt}}$
  end
end
return $\left(y_1^{\text{new}} \dots y_i^{\text{new}} \dots y_N^{\text{new}}\right)$

Multi-stage training We found that an initial period of training where the auxiliary objective is prioritized (i.e. the NAT loss is multiplied by a very small coefficient) and the DnC policy is not used, improved overall performance. Transitioning to a higher NAT loss and turning on the delete-and-copy policy later on in training increased the stability of our model. We therefore propose training NATAC models as follows:

1. **Warm-Up stage**: $\alpha = 0$ and λ are very small.
2. **Transition stage**: α increases gradually from 0 to 1, λ also increases gradually to a larger value (approximately $100\times$ larger than its initial value).
3. **Clustering stage**: $\alpha = 1$, λ is large.

Dimensionality of the latent space In all of our experiments, we found that the best performing models tend to have a latent space dimensionality between 6 and 12. At dimensionalities much larger than this, the model collapses to very few points during the transition stage of training, possibly due to the high expressiveness of the latent space. On the other hand, using a low dimensional representation results in an information bottleneck too small to sufficiently learn the auxiliary objective. For example, when clustering tweets from our Twitter dataset, a latent space of two dimensions was too small for the decoder to reliably reconstruct tweets from latent vectors. With an auxiliary objective that cannot be effectively learned, centroids collapse to a single point.

3 Experiments

We now describe the datasets and evaluation metrics used in our experiments followed by the presentation and analysis of our results in comparison to others. The full details regarding the hyperparameters used in our experiments can be found in appendix D.

Datasets We evaluate our models on four different datasets - two image and two text datasets. For images we use MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017). For text we use 20 Newsgroups (Joachims, 1996) and a Twitter dataset which we gather ourselves.

3.1 Evaluation

Our key evaluation metric is the normalized mutual information (NMI) score (Strehl & Ghosh, 2002) which measures the information gained from knowing cluster membership whilst taking the number of clusters into account. Values of NMI range from 0, where clustering gives no information, to 1, where clustering gives perfect information i.e. cluster membership is identical to class membership.

In our experiments, we train models on the concatenation of the train and validation sets and evaluate them on the test set. This is done by computing the latent representations of test examples and then assigning them to their nearest respective centroids, then computing the NMI score. Additionally,

we provide classification error scores on the MNIST dataset to compare ourselves to other related methods. The guiding motivation behind our experiments is to analyze how well our models learns cluster membership and latent representations.

3.2 MNIST

Introduced by LeCun et al. (1998), MNIST is canonical dataset for numerous machine learning tasks including clustering. MNIST has ten classes, corresponding to the ten digits 0-9, and contains 60,000 train and 10,000 test examples.

We train our model with an auxiliary reconstruction loss and use small convolutional architectures (see Figure 6) for our encoder and decoder networks. As points of comparison we also provide results of using k-means on the latent representations learned by our model (NATAC-k in Table 1) and k-means on representations learned by a simple autoencoder with the same encoder and decoder architecture (AE-k in Table 1).

Table 1 shows our model's performance is best when $d = 10$ and worse for much lower or higher values of d. This indicates that the dimensionality of the latent space impacts our model's performance (see Section 2.3 for further discussion).

The ability of our model to cluster MNIST examples well is shown by two keys results. First, it beats both NATAC-k and AE-k (Table 1). Second, it achieves state-of-the-art results when compared to other methods (see NMI column in Table 2). Note that the second result reported in Yang et al. (2016) is achieved using a separate re-clustering step after training. To re-iterate, other clustering techniques cited in Table 2 required a predefined number of clusters (in the case of MNIST k=10).

We note that NATAC-k beats AE-k, indicating that our model learns better representation as compared to a simple autoencoder. Consequently, NATAC also achieves a state-of-the-art NMI on MNIST, as measured by the classification error result (see Table 2).

Finally, we discuss the number of centroids our model converges on (see Table 1). We show that our model is successfully capable of finding centroids that represent different digits, as shown in the top row of Figure 2. However, the model also learns centroids that contain very few examples for which the decoded images do not represent any handwritten digits, as shown in the second row of Figure 2. Even with these "dead centroids", the model still performs well. Interestingly, the model also differentiates between ones with different slopes. This suggests that the latent representations of these digits are sufficiently far apart to warrant splitting them into different clusters.

3.3 FASHION-MNIST

Introduced in Xiao et al. (2017), Fashion-MNIST is a convenient swap-in dataset for MNIST. Instead of digits the dataset consists of ten different types of clothes. There are 60,000 train and 10,000 test examples just like in MNIST. Fashion-MNIST is generally considered more difficult than MNIST, with classifiers scoring consistently lower on it.[3]

The model and analysis from the previous section carry over for fashion-MNIST with a few additional important points. First, the differences between NATAC-k and AE-k are less pronounced (see Table 1) in fashion-MNIST which indicates that the representations learned by NATAC in comparison to a simple autoencoder are not as important for k-means clustering. Interestingly, our model still outperforms both NATAC-k and AE-k.

Qualitatively, Figure 2 shows that the model separates garments into slightly different categories than the labels in the dataset. For example, the most dense cluster seems to be a merging of both "pullovers" and "shirts", suggesting that the model finds it difficult to separate the two different garments. We ourselves find it difficult to discriminate between the two categories, as the low-resolution images do not easily show whether or not the garment has buttons. Additionally, the "sandal" class has been split into two separate clusters: flip-flops and high-heeled shoes with straps. This indicates that our model has found an important distinction between these two type of shoes, that the original Fashion-MNIST labels ignore. Similarly to MNIST, our model also learns "dead

[3] A MNIST vs Fashion-MNIST comparison: https://github.com/zalandoresearch/fashion-mnist#benchmark

	MNIST				Fashion-MNIST			
d	**Centroids**	**NATAC**	**NATAC-k**	**AE-k**	**Centroids**	**NATAC**	**NATAC-k**	**AE-k**
3	2251	0.455	0.455	0.452	2330	0.442	0.441	0.439
8	120	0.820	0.681	0.650	101	0.614	0.542	0.538
9	308	0.782	0.640	0.611	77	**0.638**	0.562	0.557
10	152	**0.878**	0.704	0.649	44	0.627	0.570	0.564
11	123	0.865	0.702	0.696	45	0.619	0.566	0.580
12	56	0.767	0.688	0.683	20	0.607	0.572	0.609

Table 1: NMI scores from varying the dimensionality of the latent space in the NATAC and baseline models. The baselines use k-means with the same number of clusters as the repsective NATAC model converged to. We include NATAC models with a latent dimensionality of $d = 3$, whose latent representations can be viewed without dimensionality reduction. Appendix A contains links to the visualizations hosted on the TensorFlow embedding projector.

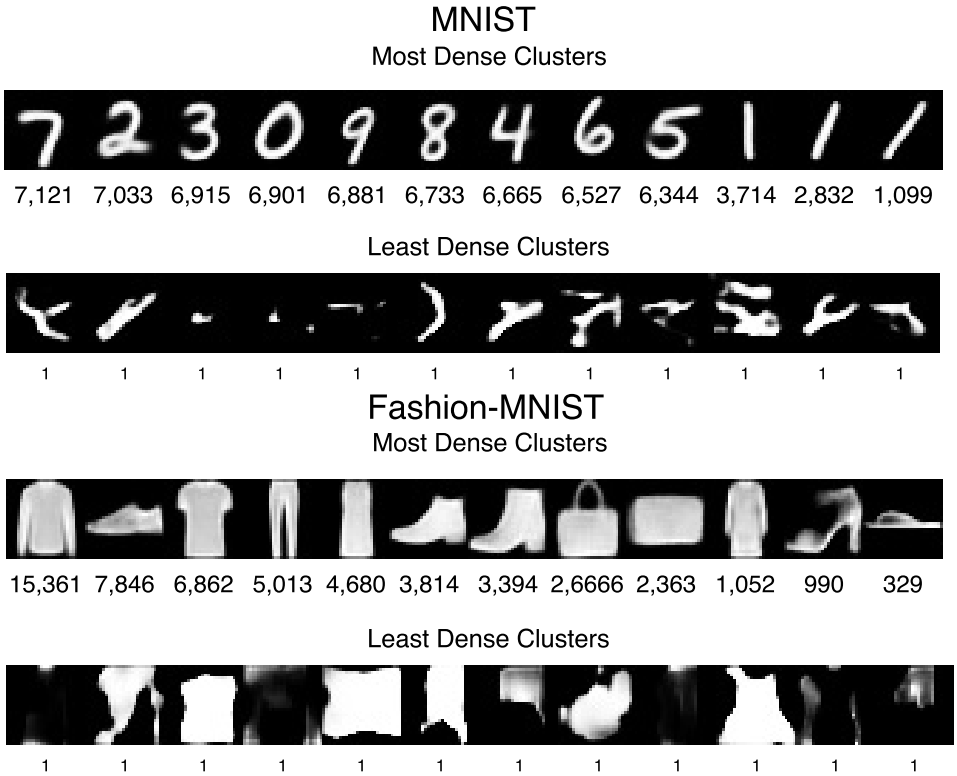

Figure 2: Outputs of the decoder when fed centroids from the MNIST experiment (top) and Fashion-MNIST experiment (bottom). The models used are those with the highest NMI, $d = 10$ and $d = 9$ for MNIST and Fashion-MNIST respectively. The top rows show decoded images from the centroids of the densest clusters (cluster sizes are shown underneath). Similarly, the bottom rows show decoded images from the least dense clusters.

clusters", which the model does not decode into any discernible garment. Further visualizations of these experiments can be found appendix section A.

3.4 Large Document Clustering on 20 Newsgroups

Introduced in Joachims (1996) the 20 Newsgroups dataset is a collection of 18,846 documents pertaining to twenty different news categories. We use the commonly used 60:40 temporal train-test

Clustering Algorithm	NMI	Classification Error (%)
k-means (MacQueen et al., 1967) On Raw Pixels	0.500	41.50
DEC (Xie et al., 2016)	-	15.70
Adversarial Autoencoders (30 clusters) (Makhzani et al., 2015)	-	4.10
Deep Gaussian Mixture VAE (Dilokthanakul et al., 2016)	-	3.08
Autencoder based clustering (Song et al., 2013)	0.669	-
Task-specific Clustering With Deep Model (Wang et al., 2016)	0.651	-
Agglomerative Clustering Using Average Linkage (Jain et al., 1999)	0.686	-
Large-Scale Spectral Clustering (Chen & Cai, 2011)	0.706	-
Yang et al.	0.906	-
Yang et al. With Re-clustering	**0.913**	-
NATAC autoencoder (Converged to 61 clusters)	0.912	**2.78**

Table 2: Comparison of our best performing NATAC model (with $d = 10$) on the entire MNIST dataset. NMI and classification error are calculated from the entire data set. We report the evaluation metric used by the authors of each respective model.

split. Interestingly, because of the temporal split in the data, the test set contains documents which differ considerably from the train set. We calculate NMI on the news categories in the dataset.

We use an auxiliary reconstruction loss and a two layer fully connected network for both the encoder and decoder, both with hidden layer sizes of 256 and ReLU nonlinearities. We represent each article as an L2 normalized term-frequency-inverse-document-frequency (TF-IDF) vector of the 5000 most occurring words in the train set.

Two common baselines are used as points of comparison for our model. A TF-IDF k-means model and a spherical k-means model. Spherical k-means is a commonly used technique of unsupervised document clustering, a good description of it can be found in Buchta et al. (2012).

Table 3 shows how the performance of the NATAC model varies with different dimensionalities of the latent space. The best NATAC models with a latent dimensionality of 3 to 7 centroids outperform a spherical k-means model with 1000 clusters, far more clusters than any of the NATAC models.

Table 3: Clustering results for the 20 Newsgroups dataset.

	NATAC		**Baselines**		
d	**Centroids**	**Test NMI**	**Method**	**Centroids**	**Test NMI**
2	190	0.260	Spherical k-means	20	0.216
3	321	0.354	Spherical k-means	50	0.178
4	328	**0.384**	Spherical k-means	100	0.306
5	217	0.375	Spherical k-means	1000	**0.337**
6	141	0.372	TF-IDF + k-means	20	0.065
7	114	0.331	TF-IDF + k-means	50	0.089
8	92	0.327	TF-IDF + k-means	100	0.198
9	46	0.293	TF-IDF + k-means	1000	0.284

3.5 Clustering Tweets

To further explore the performance of our model on text data we build a dataset of 38,309 ASCII-only tweets of English speaking users containing exactly one hashtag. The dataset has 647 different hashtags, with each hashtag having at least ten tweets containing it. We use 5000 of the tweets as the test set. As a preprocessing step, URLs and hashtags are replaced with special characters. We calculate NMI on the hashtag used in each tweet.

We train a character-based Sequence-to-Sequence autoencoder on the Twitter dataset. Just as before we use an auxiliary reconstruction loss. We set the encoder to be a bidirectional GRU with a hidden size of 64 followed by a fully connected layer which takes the GRU's final output and maps it to the latent space. The decoder uses a fully connected layer to map the latent vectors to a 128 dimensional vector which is then used as the initial hidden state for the decoder GRU.

We also experiment with using a larger encoding for both the encoder and decoder. The encoder produces an encoding with e dimensions which is then mean pooled to produce the latent representation with d dimensions. The decoder then takes the larger encoding as input.

Similarly to section 3.4, we use standard k-means and spherical k-means as baselines. As with 20 newsgroups, the NATAC model outperforms both baselines giving us an indication of its stability across different datasets.

Rather than varying the number of clusters for the baselines, we instead vary the vocabulary size for the term-frequency vectors. As the converged number of centroids for the NATAC models are very similar across different values of d (around 5000), we felt that varying the vocabulary size rather than the number of centroids would give a better range of comparable values.

As shown in table 4, spherical clustering does better on the Twitter dataset than standard k-means. The NATAC models with $d = 8, 10$ outperform all of the k-means baselines, while the lower dimensional NATAC models perform on par with spherical k-means. We see a small increase in the test NMI when using a larger encoding for the decoder and mean pooling to the latent representation.

Table 4: Clustering results for the Twitter dataset; k-means models trained with 5000 centroids.

		NATAC		**Baselines**		
d	e	**Centroids**	**Test NMI**	**Method**	**Vocab.**	**Test NMI**
4	$1 \times d$	4,733	0.753	Spherical k-means	250	0.743
6	$1 \times d$	3,838	0.742	Spherical k-means	500	**0.746**
8	$1 \times d$	4,778	0.760	Spherical k-means	1,000	0.740
10	$1 \times d$	4,000	0.759	Spherical k-means	2,500	0.723
4	$16 \times d$	2,560	0.719	TF-IDF + k-means	250	0.706
6	$16 \times d$	4,005	0.749	TF-IDF + k-means	500	0.712
8	$16 \times d$	6,343	**0.762**	TF-IDF + k-means	1,000	0.706
10	$16 \times d$	4,795	0.760	TF-IDF + k-means	2,500	0.691

4 CONCLUSION

In this paper, we present a novel neural clustering method which does not depend on a predefined number of clusters. Our empirical evaluation shows that our model works well across modalities. It improves state-of-the-art clustering performance on MNIST, without needing to receive the required number of clusters as input. Further, it outperforms powerful baselines on Fashion-MNIST and text datasets (20 Newsgroups and a Twitter hashtag dataset). However, NATAC does require some hyperparameters to be tuned, namely the dimensionality of the latent space and the function for the loss coefficient λ.

Future work Several avenues of investigation could flow from this work. Firstly, the effectiveness of this method in a semi-supervised setting could be explored using a joint reconstruction and classification auxiliary objective. Another interesting avenue to explore would be different agglomerative policies other than delete-and-copy. Different geometries of the latent space could also be considered other than a unit normalized hypersphere. To remove the need of setting hyperparameters by hand, work into automatically controlling the coefficients (e.g. using proportional control) could be studied. Finally, it would be interesting to see whether clustering jointly across different feature spaces would help with learning better representations.

REFERENCES

Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017.

Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012.

Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, volume 5, pp. 14, 2011.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. Wiley, New York, 1973.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pp. 226–231, 1996.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

John C Gower. A comparison of some methods of cluster analysis. *Biometrics*, pp. 623–637, 1967.

Bock Hans-Hermann. Origins and extensions of the k-means algorithm. *J. EHPS*, 4(2), 2008.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *CSUR*, 1999.

Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Harold W Kuhn. The hungarian method for the assignment problem. *NRL*, 2(1-2):83–97, 1955.

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

Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, pp. 5076–5084, 2016.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *SMSP*, volume 1, pp. 281–297. Oakland, CA, USA., 1967.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Lt Orloci. An agglomerative method for classification of plant communities. *J.Ecol*, 1967.

RS Sokol and Peter HA Sneath. Principles of numerical taxonomy, 1963.

Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pp. 117–124. Springer, 2013.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Zhangyang Wang, Shiyu Chang, Jiayu Zhou, Meng Wang, and Thomas S Huang. Learning a task-specific deep architecture for clustering. In *ICDM*, pp. 369–377. SIAM, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pp. 478–487, 2016.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pp. 5147–5156, 2016.

Appendix

A VISUALIZATIONS OF MNIST/FASHION-MNIST LATENT REPRESENTATIONS

We have published the visualizations of MNIST/Fashion-MNIST representations on the TensorFlow embedding projector. We recommend selecting the "Spherize data" option, as this gives a much clearer view of the embeddings. The images used are taken from the test set of MNIST and Fashion-MNIST respectively. We include the NATAC models with the highest NMI from each dataset, along with the models with $d = 3$, which require no dimensionality reduction to visualize.

- MNIST $d = 3$:
  http://projector.tensorflow.org/?config=https://raw.githubusercontent.com/neuralclusteringNAT/paper-resources/master/mnist_embeddings/digits3/config.json
- MNIST $d = 10$:
  http://projector.tensorflow.org/?config=https://raw.githubusercontent.com/neuralclusteringNAT/paper-resources/master/mnist_embeddings/digits10/config.json
- Fashion-MNIST $d = 3$:
  http://projector.tensorflow.org/?config=https://raw.githubusercontent.com/neuralclusteringNAT/paper-resources/master/mnist_embeddings/fashion3/config.json
- Fashion-MNIST $d = 9$:
  http://projector.tensorflow.org/?config=https://raw.githubusercontent.com/neuralclusteringNAT/paper-resources/master/mnist_embeddings/fashion9/config.json

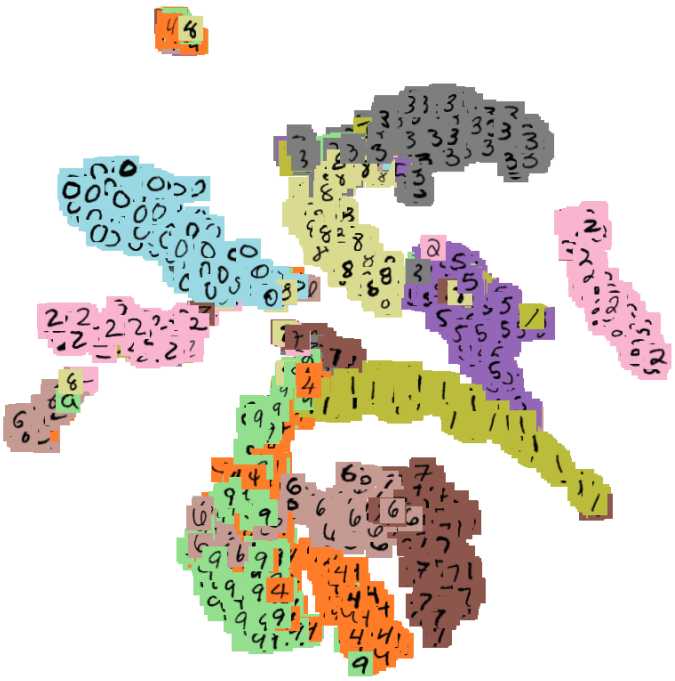

Figure 3: T-SNE visualization of the best performing NATAC model for MNIST $d = 10$

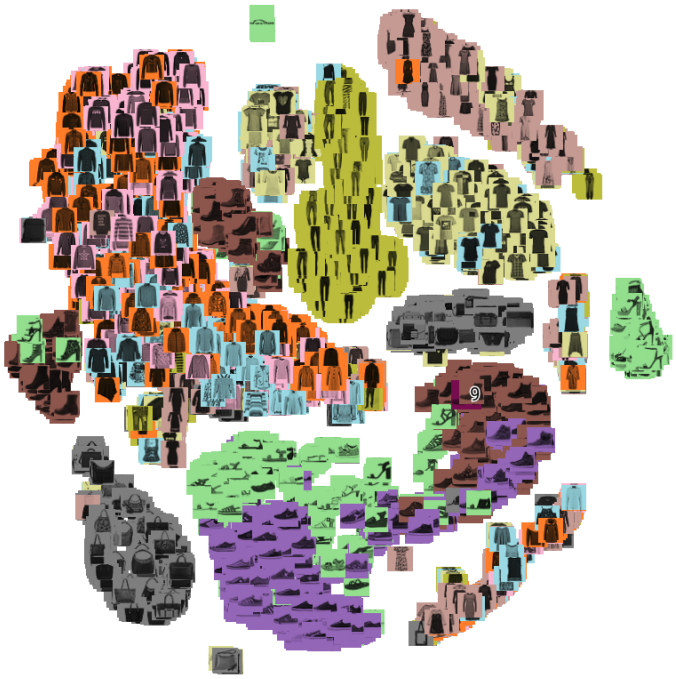

Figure 4: T-SNE visualization of the best performing NATAC model for Fashion-MNIST $d = 9$

B EXAMPLES FROM THE FASHION-MNIST DATASET.

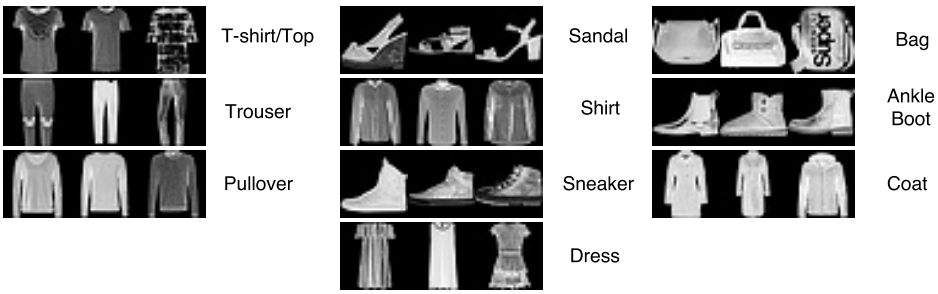

Figure 5:

C ADDITIONAL DISCUSSION ON TRAINING NATAC MODELS

C.1 GEOMETRY OF LATENT SPACE

We experimented with using polar coordinates early on in our experiments. Rather than using euclidean coordinates as the latent representation, z is considered a list of angles $\theta_1, \theta_2 \cdots \theta_n$ where $\theta_1 \cdots \theta_{n-1} \in [0, \pi]$ and $\theta_n \in [0, 2\pi]$. However, we found that the models using polar geometry performed significantly worse than those with euclidean geometry.

Additionally, we also experimented with not L2 normalizing the output of the encoder network. We hypothesized that the model would learn a better representation of the latent space by also "learning" the geometry of the noise targets. Unfortunately, the unnormalized representation caused the noise targets to quickly collapse to a single point.

D HYPERPARAMETERS FOR EXPERIMENTS

Although each different modality (monochrome images, bag-of-words, sequence of characters) uses a different set of hyperparameters, we follow a similar recipe for determining the values for each one:

- We use a large batch size of 100. This is so each batch has a representative sample of the targets to reassign in each training step.
- The warm-up period is calculated by observing when the auxiliary objective starts to converge.
- The final value for λ in training is set so that the NAT loss was approximately 1% of the total loss.
- The initial value for λ is set as approximately 1% of the final value of λ.
- The transition phase typically lasts 100 epochs of training.
- During the transition phase, the value of α is incrementally increased from 0 to 1.

We now explicitly list the hyperparameters used for each experiment:

D.1 MNIST AND FASHION MNIST

- A batch size of 100.
- A warm-up period of $10 \times d$ epochs, during which $\lambda = 0.001$.
- A transition period lasts for 250 epochs, where λ is incrementally increased to 0.25, and α is incremented from 0 to 1.
- The ADAM optimizer (Kingma & Ba, 2014) with a learning rate ($\alpha = 10^{-5}$)

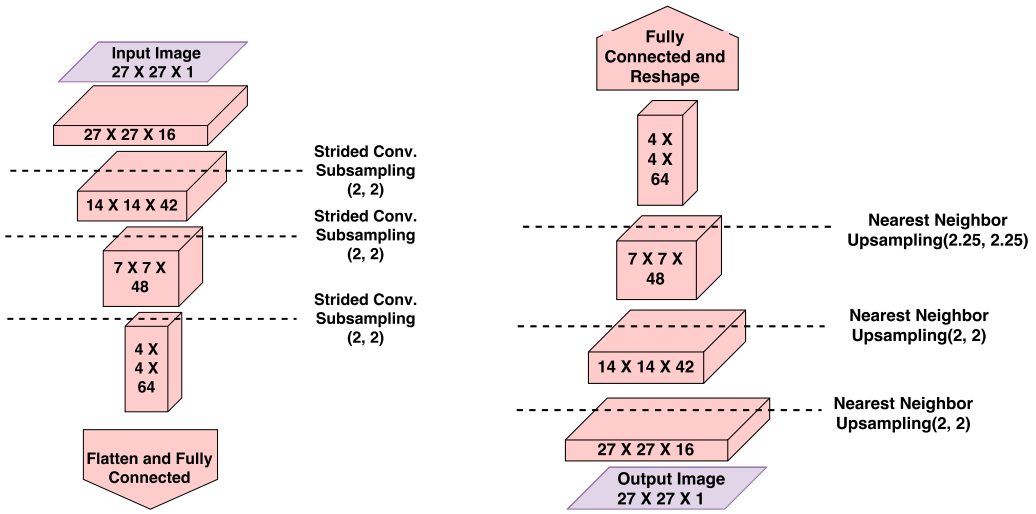

Figure 6: Architecture of the encoder (left) and decoder (right) used for the MNIST experiments. Between each subsampling layer in the encoder, a single convolution layer is applied with a filter shape of 3×3 with border padding to keep the same shape before and after the convolution. Similarly, in the decoder one transpose convolutional layer is applied between each upsampling layer, 3×3 filter shape and shape-preserving padding.

D.2 20 NEWSGROUPS

- A batch size of 100.
- A warm-up period of 100 epochs, during which 10^{-4}.
- A transition period lasts for 100 epochs, where λ is incrementally increased to 0.01, and α is incremented from 0 to 1.

- The ADAM optimizer (Kingma & Ba, 2014) with a learning rate ($\alpha = 10^{-5}$).
- Dropout with a keep-probability of 0.95 in the hidden layers of the encoder and decoder.

D.3 TWITTER DATASET

- A batch size of 100.
- A warm-up period of 100 epochs, during which $\lambda = 0.001$.
- A transition period lasts for 100 epochs, where λ is incrementally increased to 0.1, and α is incremented from 0 to 1.
- The ADAM optimizer (Kingma & Ba, 2014) with a learning rate ($\alpha = 10^{-5}$).