# Implementation and Improvements to Deep Dictionary Model

Chen-Han Tsai[1] (985940600) and Bojun Ouyang[1] (973959174)

School of Electrical Engineering, Tel Aviv University, Israel
{chenhantsai, bojungouyang}@mail.tau.ac.il

**Abstract.** In this project, we aim to investigate the topic of Deep Dictionary Learning by combining well known Deep Learning techniques. We first re-implement the proposed algorithm by Tariyal *et al* [7] using the PyTorch framework. We then extend upon their work by suggesting an improved classifier, and we demonstrate such an improvement on several well known datasets. The code following this project is available at `https://github.com/mxtsai/deep_dictionary_learning`

**Keywords:** Deep Dictionary Learning

## 1 Deep Dictionary Learning

### 1.1 Sparse Dictionary Learning

Sparse Dictionary Learning is a method of learning signal representations and their corresponding basis (referred to as "atoms") for which they are constructed from. The observed signal $x \in R^d$ is assumed to be the output of a generative model that constructs such signals by forming linear combinations of atoms (dictionary elements) from a dictionary $D \in R^{d \times n}$.

When $D$ is known, this problem is considered a sparse coding problem, and the goal is to find a representation $\gamma \in R^n$ such that

$$x = D\gamma \tag{1}$$

is satisfied for the observed signal $x$. This is done by optimizing the following objective

$$\min_{\gamma \in R^n} \|\gamma\|_0 \text{ subject to } \|x - D\gamma\|_2^2 \leq \epsilon^2 \tag{2}$$

where $\epsilon$ is to account for the noisy signal observed, and the $l_0$ norm is the count for the number of non-zero values in the representation $\gamma$. Since the atoms might not necessarily be orthogonal and $D$ might be an over-complete spanning set, it is often required that the representation $\gamma$ to be sparse.

Dictionaries with $n > d$ (number of atoms greater than dimension of each atom) are referred to as "overcomplete" whereas dictionaries with $n < d$ are referred to as "undercomplete". In this project, we focus on the undercomplete case since it is similar to learning a lower dimension representation of the signal (e.g. Principal Component Analysis [1]) which is more applicable to later tasks such as classification.

The objective listed in Equation 2 is an NP-Hard problem, and therefore, approximate solutions are can be computed using some well known algorithms such as Matching Pursuit [4] and Basis Pursuit [6].

Matching Pursuit is a greedy algorithm that performs dot product between all the atoms in the dictionary and the observed signal $x$ to construct the representation $\gamma$. The coefficient in $\gamma$ corresponding to the atom with the highest dot product value is first registered, and a residual vector $R$ is computed by subtracting from $x$ the atom with the highest dot product value. This process is then continuously performed on the residual $R$, and the coefficients in $\gamma$ are updated until a stopping criterion has been reached. Since the atom selected during each iteration is based on the dot product with the residual $R$, it is possible that an atom will be selected multiple times.

To mitigate this effect, the Orthogonal Matching Pursuit was proposed, and it solves this issue by updating the current coefficients in $\gamma$ by calculating the orthogonal projection of $x$ spanned by the current selected atoms at the current iteration. Details on the implementation is available in [5].

Basis Pursuit is another approach that replaces the $l_0$ norm with $l_1$ norm, and solves for

$$\min_{\gamma \in R^n} \|\gamma\|_1 \;\text{ subject to }\; \|x - D\gamma\|_2^2 \leq \epsilon^2 \tag{3}$$

instead. The issue with the $l_0$ norm is that the objective is not differentiable, and by relaxing the problem (with a cost of not obtaining an exact solution) with the $l_1$ norm, the problem becomes a convex problem that can be solved with general convex solvers.

The aforementioned techniques are suitable for problems where the dictionary $D$ is know. In the general sense, however, the dictionary $D$ is unknown and one would have to solve for $D$ as well as a sparse representation $\gamma$ for $x$. This problem is known as the Sparse Dictionary Learning problem, and we formulate this problem for a given set of observed signals.

For a set of observed signals $X = [x_1, x_2, ..., x_m],\ x_i \in R^d$, some initial dictionary $D \in R^{d \times n}$, and initial representations $\Gamma = [\gamma_1, \gamma_2, ..., \gamma_m],\ \gamma_i \in R^n$, we aim to optimize the following objective

$$\arg\min_{D,\Gamma} \sum_{i=1}^{m} \|x_i - D\gamma_i\|_2^2 + \lambda\|\gamma_i\|_0 \tag{4}$$

where $\gamma$ is a hyperparameter that determines the tradeoff between the sparseness of the representation and the reconstruction $D\Gamma$. One of the most well known techniques to solve the Sparse Dictionary Learning problem is K-SVD [2]

K-SVD is a generalization of K-means clustering, and the algorithm works by iteratively fixing $D$ and updating $\Gamma$, then fixing $\Gamma$ and updating $D$, and this repeated until convergence. The step where $D$ is fixed is the sparse coding problem, and one may solve the optimal $\Gamma$ for the fixed $D$ using the techniques we mentioned above. The next step is the dictionary learning step, and each atom $d_i$ in $D$ is update one at a time. During each atom update step, a residual $R$ is computed by subtracting the contribution of all other atoms $d_{j\neq i}$ from the observations in $X$ that have a contribution from $d_i$ (which is being updated). A new atom $\hat{d}_i$ then computed to maximize the contribution of this atom for the residual $R$ using SVD. This process is repeated for each atom in the dictionary, and once the dictionary has been updated, the sparse coding problem is then repeated using this new dictionary.

Other solutions to the Sparse Dictionary Learning problem include Stochastic Gradient Descent and other iterative methods. Stochastic Gradient Descent relaxes the $l_0$ norm by replacing it with the $l_1$ norm,

$$\arg\min_{D,\Gamma} \sum_{i=1}^{m} \|x_i - D\gamma_i\|_2^2 + \lambda\|\gamma_i\|_1 \tag{5}$$

and a random subset of observations $C$ from $X$ are selected to compute the gradients of the reconstruction loss with respect to $D$. $D$ is then updated by

$$D_{j+1} = D_j - \eta_D \cdot \nabla_D \left( \sum_{i\in C} \|x_i - D\gamma_i\|_2^2 + \lambda\|\gamma_i\|_1 \right) \tag{6}$$

for which $\eta_D$ is the gradient update step size for the dictionary learning step. Using this updated dictionary, one may solve the sparse coding problem using the mentioned techniques.

### 1.2 Deep Dictionary Learning

In the paper by Tariyal *et al*, the Deep Dictionary model was proposed [7] by extending Sparse Dictionary Learning in a layer-by-layer fashion similar to the design of deep neural networks. The Deep Dictionary model is defined by a series of *dictionary layers* denoted as $\{l_i\}_{i=1}^{L}$. Each *dictionary layer* $l_i$ stores a

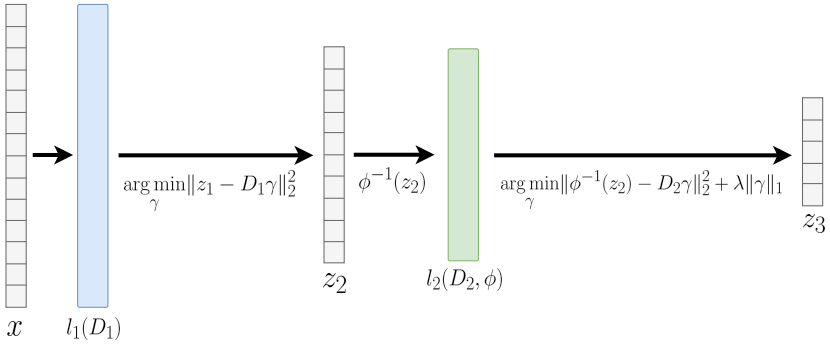


**Fig. 1.** Example of a 2 layer Deep Dictionary Model performing layer-wise sparse coding of $x$ (dictionary is kept constant during sparse coding)

per-layer undercomplete dictionary $D_l$ for which we can compute a sparse representation $z_{l+1}$ from a given input $z_l$ coming from layer $l$. For a fixed dictionary $\Omega = \{D_1, D_2, ..., D_L\}$ and an input $x$ (which is $z_1$), we can compute $z_2$ by solving the sparse coding problem:

$$z_2 = \arg\min_{\gamma} \|z_1 - D_1\gamma\|_2^2 \tag{7}$$

In the general sense, the sparse representation $z_l$ of layer $l$ can be computed by solving

$$z_l = \arg\min_{\gamma} \|\phi^{-1}(z_{l-1}) - D_{l-1}\gamma\|_2^2 \quad \forall l \in \{1, 2, ..., L\} \tag{8}$$

where $\phi$ is an activation function. In the paper by Tariyal *et al*, only the identity function was explored. In our project, we experiment with additional non-linear actuation's and compare their differences.

For the final layer, sparseness is explicitly enforced by adding the sparseness objective in addition to the reconstruction loss (previous layers only enforced reconstruction loss):

$$z_{L+1} = \arg\min_{\gamma} \|\phi^{-1}(z_L) - D_L\gamma\|_2^2 + \lambda\|\gamma\|_1 \tag{9}$$

An example of a two layer Deep Dictionary model ($L = 2$) performing layer-wise sparse coding of $z_2$ and $z_3$ of the input $x$ is displayed in Figure 1.

To perform layer-wise sparse dictionary learning (learning the layer wise dictionaries alongside the sparse representations), the authors suggested the following algorithm

**Algorithm 1:** Training Algorithm for the Deep Dictionary Model

**Input:** Observed signal $x$, Spareness coefficient $\lambda$
**Output:** A trained Deep Dictionary model
**Initialize:** *dictionary layers* $\Omega = \{D_1, D_2, ..., D_L\}$
**for** $l$ *in* {*1, ..., L*} **do**
  **if** $l = 1$ **then**
    **while** *not converged* **do**
      $\mathrm{z}_2 = \arg\min_{\gamma} \|z_1 - D_1\gamma\|_2^2$
      $\mathrm{D}_1 = \arg\min_{D} \|z_1 - Dz_2\|_2^2$
    **end**
  **else if** $l \in \{2, ..., L-1\}$ **then**
    **while** *not converged* **do**
      $\mathrm{z}_{l+1} = \arg\min_{\gamma} \|\phi^{-1}(z_l) - D_l\gamma\|_2^2$
      $\mathrm{D}_l = \arg\min_{D} \|\phi^{-1}(z_l) - Dz_{l+1}\|_2^2$
    **end**
  **else**
    **while** *not converged* **do**
      $\mathrm{z}_{L+1} = \arg\min_{\gamma} \|\phi^{-1}(z_L) - D_L\gamma\|_2^2 + \lambda\|\gamma\|_1$
      $\mathrm{D}_L = \arg\min_{D} \|\phi^{-1}(z_L) - Dz_{L+1}\|_2^2$
    **end**
  **end**
**end**

The algorithm described in Algorithm 2 trains the Deep Dictionary model in a layer by layer fashion. Each layer is optimized by iterative training, starting with $l_1$. During the optimization of each layer, the representation $z$ is first optimized, then subsequently the dictionary $D$ is then optimized, and this procedure is repeated until convergence. Once the first layer has converged, the same iterative training for the second layer begins, and this is repeated across all the layers. At the end, we obtain a trained Deep Dictionary Model.

Following a trained Deep Dictionary model, the authors proceeded by training a classifier using the extracted sparse representation $z_{L+1}$ as input. The inputs to their Deep Dictionary model were categorical data (e.g. MNIST), and the idea was that the final sparse representation $z_{L+1}$ contains enough encoded information to be used for classifying the categories for which these inputs belong to. They experimented with Support Vector Machines, Stacked Auto-Encoders, and Deep Belief Networks, and the best classification performance was mostly observed with the Support Vector Machine. In this project, we extend upon their work by introducing the Multi-Layer Perceptron as the classifier. We demonstrate the effectiveness of the Multi-Layer Perceptron performing classification tasks on a wider range of categorical datasets. In the next section, we explain

in detail how we re-implemented the Deep Dictionary model using the PyTorch framework, and we explain in depth how the optimization is performed.

### 1.3 Implementation Details

Following the suggested model and training algorithm proposed by Tariyal *et al*, we re-implemented the algorithm in the PyTorch framework. We detail the specifics of our implementation, and we suggest some improvements. We first consider the case of a single observed signal, and then we extend the solution to a batch of observed signals.

For a single observed signal $x \in R^d$ (equals $z_1$), we have the corresponding dictionary $D_1 \in R^{n_1 \times d}$ ($n_1$ is the number of atoms for $D_1$) at layer $l$. To find the optimal $z_2 \in R^{n_1}$, we solve for

$$z_2 = \arg\min_{\gamma} \|z_1 - D_1\gamma\|_2^2 \tag{10}$$

using Least Squares. Since the objective listed in Equation 10 is convex, we can take the gradient of the norm with respect to $\gamma$, require that it equals zero, and solve for the optimal $\gamma$. Therefore, solving the optimal $\gamma$ yields

$$z_2 = (D_1 D_1^T)^{-1} D_1 z_1 \tag{11}$$

Similarly, we can solve for the optimal $D_1$ using Least Squares and we obtain:

$$D_1 = (z_2 z_2^T)^{-1} z_2 z_1^T \tag{12}$$

In a similar manner, we can extend the above formulations to a batch of observed signals. For $n$ observed signals $\{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$, we can construct $X \in R^{n \times d}$ (equals $Z_1$) as

$$X = \begin{bmatrix} -x^{(1)T}- \\ \vdots \\ -x^{(n)T}- \end{bmatrix} = Z_1 \tag{13}$$

and the update step for obtaining $Z_2 \in R^{n \times n_1}$ is :

$$Z_2 = \begin{bmatrix} -z^{(1)T}- \\ ... \\ -z^{(n)T}- \end{bmatrix} = Z_1 D_1^T (D_1 D_1^T)^{-1} \tag{14}$$

and the update step for $D_1$ is:

$$D_1 = (Z_2^T Z_2)^{-1} Z_2^T Z_1 \tag{15}$$

In a similar manner, we can extend the above formulations to the training algorithm from Algorithm 2. Our training algorithm is listed as follows:

**Algorithm 2:** Alternating Least Squares Optimization of the Deep Dictionary Model

**Input:** Batch of signals $X$, Spareness coefficient $\lambda$
**Output:** A trained Deep Dictionary model
**Initialize:** *dictionary layers* $\Omega = \{D_1, D_2, ..., D_L\}$
**for** $l$ *in* {*1, ..., L*} **do**
  **if** $l = 1$ **then**
    **while** *not converged* **do**
      $\mathrm{Z}_2 = Z_1 D_1^T (D_1 D_1^T)^{-1}$
      $\mathrm{D}_1 = (Z_2^T Z_2)^{-1} Z_2^T Z_1$
    **end**
  **else if** $l \in \{2, ..., L-1\}$ **then**
    **while** *not converged* **do**
      $\mathrm{Z}_{l+1} = \phi^{-1}(Z_l) D_l^T (D_l D_l^T)^{-1}$
      $\mathrm{D}_l = (Z_{l+1}^T Z_{l+1})^{-1} Z_{l+1}^T \phi^{-1}(Z_l)$
    **end**
  **else**
    **while** *not converged* **do**
      $\mathrm{z}_{L+1} = \phi^{-1}(Z_L) D_L^T (D_L D_L^T + \lambda I)^{-1}$
      $\mathrm{D}_L = (Z_{L+1}^T Z_{L+1})^{-1} Z_{L+1}^T \phi^{-1}(Z_L)$
    **end**
  **end**
**end**

By optimizing the model in batch, we are able to train a Deep Dictionary Model in a matter of minutes when just running from CPU. Next, we present an MLP classifier that aims to classify categorical data for a given input $x$.

The MLP classifier we propose is a simple 2 layer MLP. The input dimension of the MLP classifier is $n_{mlp}$, the dimension of the first and second hidden layer is $n_{mlp}/2$ and $n_{mlp}/4$, and the dimension of the output layer is the number of classes to classify for. The non-linear activation is the Sigmoid activation 16

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{16}$$

and the final class probabilities are computed using the Softmax activation 17

$$\text{Softmax}(v)_i = \frac{\exp(v_i)}{\sum_{j=1}^{C} \exp v_j} \tag{17}$$

In the original paper, the authors suggest that the input to the classifier (e.g. SVM) is the final sparse representation $z_{L+1}$. However, we prose an alternate input to the classifier that preserves information across the layers. We suggest to concatenate the representations across all the *dictionary layers* and feed this vector $z_c$ as the input to the classifier.

$$z_c = [-z_2-, ..., -z_L-, -z_{L+1}-]^T \in R^{n_1+...+n_L} \tag{18}$$

By feeding $z_c$, we incorporate both low-level information (earlier representations) and high-level information (later representations) into the classifier, and in our experiments, we show that the classifier's performance remains consistent despite varying depths when using $z_c$ instead of just $z_{L+1}$.

Lastly, we train the MLP using standard Cross Entropy Loss

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log\left(p_{i,c}\right) \cdot \mathbb{1}[y_i = c] \tag{19}$$

and the MLP is optimized using variants of Stochastic Gradient Descent. In the next few sections, we demonstrate our implementation of the Deep Dictionary model on several datasets, and we compare our proposed algorithm of classifying these categorical data compared with the original approach.

## 2 Experiments

### 2.1 Datasets

**Fashion MNIST.** Fashion MNIST is a dataset that contains 70,000 (60,000 for training, 10,000 for test) grayscale images of dimension $28 \times 28$ depicting categories of clothing items [8]. Some of the categories include trousers, pullovers, dresses, handbags, and boots, and the images are uniformly distributed across each category.

**EMNIST.** EMNIST is a dataset containing $28 \times 28$ grayscale images of handwritten characters derived from the NIST Special Database 19 [3]. There are several variations of the EMNIST dataset, and in this work, we focus on EMNIST-Balanced version of the dataset. EMNIST-Balanced contains 131,600 images from 47 different characters, and the class distribution is balanced all across.

### 2.2 Varying Depth and Activation Functions

In this section, we compare the effects of how different Deep Dictionary Model activation functions affect the performance of the entire model. The first sub-task

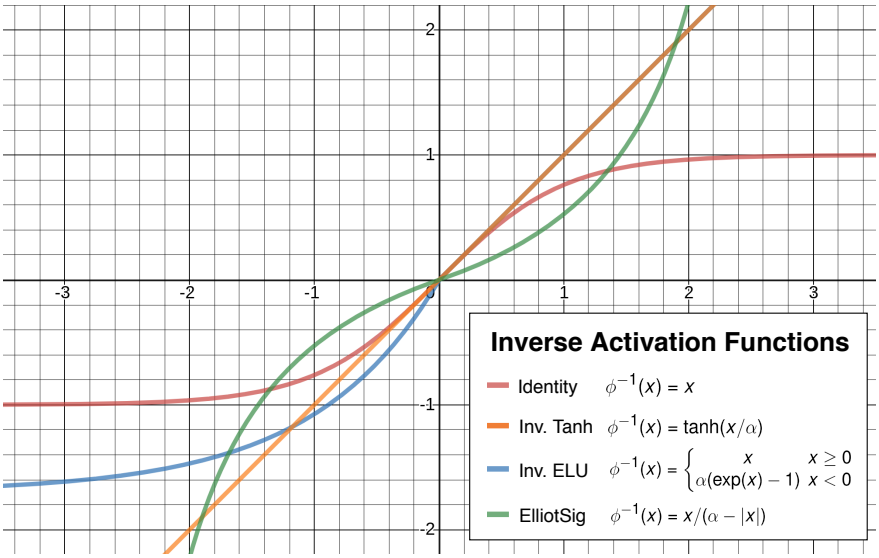


**Fig. 2.** Graph of the respective inverse activation functions.

is to reconstruct the original image, and we are interested in knowing how different activation functions affect the reconstructed images. The second subtask is to investigate how the sparse representations $z_c$ generated from the different activation functions affect the MLP classification performance.

The experiments are carried out using the Fashion MNIST dataset. The input dimension is 768, and each additional layer in the Deep Dictionary model halves the dimension of the previous layer (e.g. a Deep Dictionary model with 2 layers contain dictionary dimension $n_1 = 384$ and $n_2 = 192$). The sparse coefficient $\lambda$ is set to 5e-1, and each layer of the Deep Dictionary Model is trained for 30 epochs with a batch size of the entire dataset. When training the MLP, we use a batch size of 2000, and we train for 10 epochs with a learning rate of 7e-3 using the RMSProp optimizer.

Experiments were performed using the Identity activation, the Tanh Inverse activation, the ELU Inverse activation, and the ElliotSig activation (see Figure 2). **(The choice of the activation function $\phi$ must be an injective function since its inverse $\phi^{-1}$ is required. In most cases, it's often better to work with the inverse of these well established activation functions so that the loss term doesn't "blow up" when training the Deep Dictionary model.)** We first train using the Identity activation and for a varying Deep Dictionary depth from $L = 1, 2, 3$. The depth of the dictionary is then fixed ($L = 3$) and we compare the effects of the different Deep Dictionary activation functions.

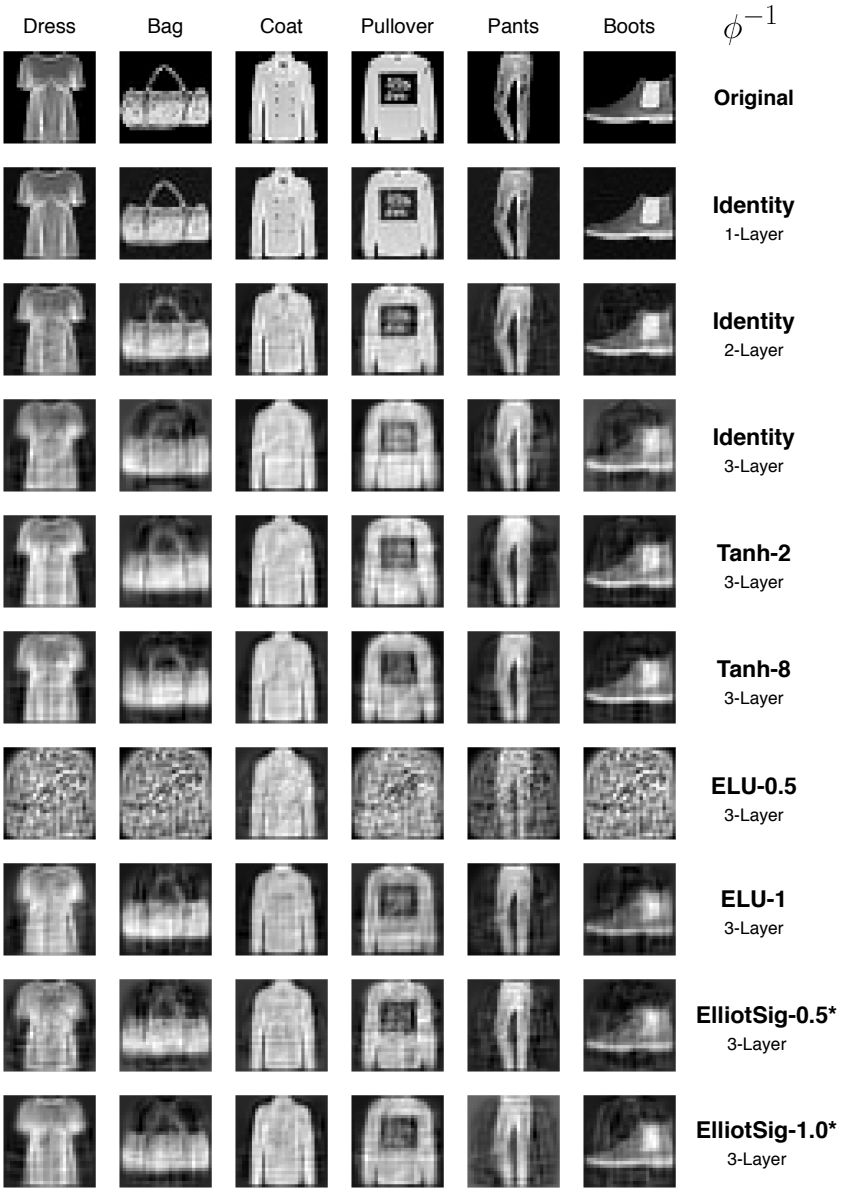


**Fig. 3.** A visualization of the reconstructions for Deep Dictionary models with varying depth and activation functions. Deep Dictionary models with $L = 1, 2, 3$ were trained using the Identity activation, and we observe some artifacts as the depth of the model increases. We then kept the depth of the network constant ($L = 3$), and we vary the activation functions. It can be observed that by changing the activation function, we able to obtain reconstructions with less distortion (e.g. Tanh).

| Activation | Alpha | Recons. Loss | Classification Loss | Accuracy |
|---|---|---|---|---|
| Identity ($L = 1$) | - | 1.044 | 0.375 | 0.868 |
| Identity ($L = 2$) | - | 4.727 | 0.337 | 0.870 |
| Identity | - | 24.492 | 0.357 | 0.871 |
| Inv. Tanh | 2 | 10.712 | 0.3822 | 0.864 |
| | 4 | 10.069 | 0.365 | 0.869 |
| | 6 | 13.433 | 0.362 | 0.871 |
| | 8 | **9.803** | 0.367 | 0.868 |
| Inv. ELU | 0.25 | 19875.52 | 0.3662 | 0.870 |
| | 0.5 | 6649.12 | 0.364 | 0.873 |
| | 1 | 16.499 | 0.361 | 0.871 |
| ElliotSig | 0.5 | 40.9315 | **0.3493** | **0.875** |
| | 0.75 | 21.58 | 0.359 | 0.872 |
| | 1 | 13.982 | 0.363 | 0.873 |

**Table 1.** Comparison of the reconstruction and classification performance for varying Deep Dictionary activation functions. Aside from the first two experiments with the Identity activation, all other Deep Dictionary models contain 3 dictionary layers.

Some sample reconstructions are shown in Figure 3, and the classification performance is listed in Table 1. As we can see from the reconstruction, the deeper the Deep Dictionary Model is, the more artifacts from other image categories tend to show. One of the reasons causing these artifacts is because of the imperfect reconstruction between layers. For example, $\hat{z_1} = D_1 z_2$ is an imperfect reconstruction of $z_1$, and we can see this error for the 1-layer Deep Dictionary model with the Identity activation. When deeper layers are then constructed, we would obtain imperfect reconstructions $\hat{z}_l = D_l z_{l+1}$ between layers $l + 1$ and $l$, and this imperfect reconstruction will lead to even more imperfect reconstructions for $z_i$ where $i < l$. Therefore, it is reasonable to expect more distorted image reconstructions with deeper Deep Dictionary models.

We then tried a variety of activation functions to see if the images can be better reconstructed despite being a 3-layer Deep Dictionary model. Through our experiments, we demonstrate that it is possible to obtain a better reconstructed image by replacing the Identity activation with other activation functions. However, the optimal choice activation function (and it's corresponding $\alpha$ parameter) requires careful tuning, and we demonstrate this with the Inverse Tanh, the Inverse ELU, and the ElliotSig activation functions. From Table 1 we can see that the best reconstruction comes from an Inverse Tanh activation with $\alpha = 8$ despite being a 3 layer Deep Dictionary model.

We then investigate how these variations in activation function lead to a difference in classification performance. From our experiments, we observe that **a good reconstruction performance of a Deep Dictionary model does not imply a better classification performance when using the MLP model.** We can see that the best activation function for performing reconstruction is the Inverse Tanh activation function as it has the least average reconstruction loss

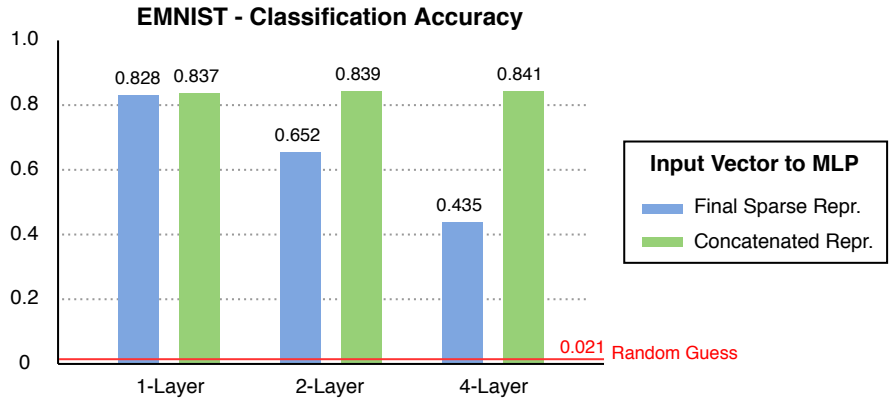


**Fig. 4.** Comparison of the MLP classification performance on the EMNIST dataset. The input to the MLP model differ between the original $z_{L+1}$ vector and our proposed $z_c$ vector. As we can see, classification performance remains consistent despite the varying number of layers of the Deep Dictionary model when we use the $z_c$ vector as input to the MLP. Performance deteriorates as the number of *dictionary layers* increase when the input to the MLP is the originally proposed $z_{L+1}$ vector.

for a wide variety of $\alpha$'s. On the other hand, the Inverse ELU and the ElliotSig activation functions do not perform as well on the reconstruction, but the average classification performance for the varying $\alpha$'s are slightly better compared with the average classification performance of the Inverse Tanh. Therefore, the optimal choice of the activation function depend on the desired task to be achieved, and in this project, we extend upon the work of the authors by suggesting several plausible activation functions.

### 2.3 Improved Classification with Concatenated Representations

In this section, we are interested in whether our proposed concatenated vector $z_c$ provides a more consistent classification performance than the sparse representation $z_{L+1}$ suggested by the original authors when used as input to the MLP. We use the EMNIST dataset for this task since it contains much more image categories than the datasets. We run the same experiments using the ElliotSig activation for Deep Dictionary models with depths of $1, 2, 4$.

The Deep Dictionary models with 1,2 and 4 dictionary layer(s) have *dictionary layer* configurations of $(n_1 = 392)$, $(n_1 = 392, n_2 = 196)$, and $(n_1 = 392, n_2 = 196, n_3 = 130, n_4 = 98)$. We train the MLP using the Adam optimizer with a learning rate of 5e-3 and a batch size of 500. The results of this experiment is presented in Figure 4.

As we can see, the amount of information that is retained at the deepest layer of the Deep Dictionary model is often not optimized for classification even

though they contain enough information for reconstructing the original image. Therefore, by concatenating the representations from earlier *dictionary layers*, we can retain more information that is useful for classification, and we demonstrate this in this experiment.

## 3 Conclusions

In this project, we re-implemented the Deep Dictionary model proposed by Tariyal *et al* using the PyTorch framework, and we provide the in-depth implementation details for how our implementation works. The original paper evaluated the Deep Dictionary model primarily on the MNIST dataset (and some of its minor variations), and in this project, we explore several other datasets for which the Deep Dictionary model may be applied.

We performed a series of experiments on the MNIST dataset by observing the reconstruction and classification performance when the number of Deep Dictionary layers are varied. We then suggested several alternate activation functions in addition to the Identity activation (which is the only activation function explored by the authors) that demonstrate improved reconstruction and classification performance compared with the Identity activation. With the observed experimental results, we show that a good reconstruction performance does not necessarily imply a good classification performance. Therefore, the optimal choice of the activation function $\phi$ depends on the desired task, and we suggest the Inverse Tanh activation for reconstruction and the ElliotSig activation for classification.

We also suggest an improved representation vector $z_c$ as the input to the MLP when the task is to classify the category for which the observed signal belongs to. In the original paper, the authors suggest to feed the representation $z_{L+1}$ obtained from the final *dictionary layer* as the input to a classifier (e.g. SVM, DBN, SAE), and in this project, we suggest a simple two layer MLP as the classifier for which the input is concatenation of the representations obtained from across all *dictionary layers*. As demonstrated in the experiments, our approach of using a concatenated representation $z_c$ enables our classifier to perform consistently despite varying Deep Dictionary depths (with slight performance gains as the Deep Dictionary model deepens).

Future works may include extending the Deep Dictionary model to multi-channel signals (e.g. RGB images). It would also be interesting to apply dictionary learning to convolutional filters, as it would allow the learning of local spatial representations that might yield better results for images with more contents and larger resolutions. We hope this project may provide some insights to the Deep Dictionary model. Our code is available at `https://github.com/mxtsai/deep_dictionary_learning` for reference.

## References

1. Abdi, H., Williams, L.J.: Principal component analysis. WIREs Computational Statistics **2**(4), 433–459 (2010). https://doi.org/10.1002/wics.101, `https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101`
2. Aharon, M., Elad, M., Bruckstein, A.: K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing **54**(11), 4311–4322 (2006)
3. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: an extension of mnist to handwritten letters (2017)
4. Mallat, S.G., Zhifeng Zhang: Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing **41**(12), 3397–3415 (1993)
5. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers. pp. 40–44 vol.1 (1993)
6. Shaobing Chen, Donoho, D.: Basis pursuit. In: Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers. vol. 1, pp. 41–44 vol.1 (1994)
7. Tariyal, S., Majumdar, A., Singh, R., Vatsa, M.: Deep dictionary learning. IEEE Access **4**, 10096–10109 (2016)
8. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)