LDMNet: Low Dimensional Manifold Regularized Neural Networks

Wei Zhu

Duke University

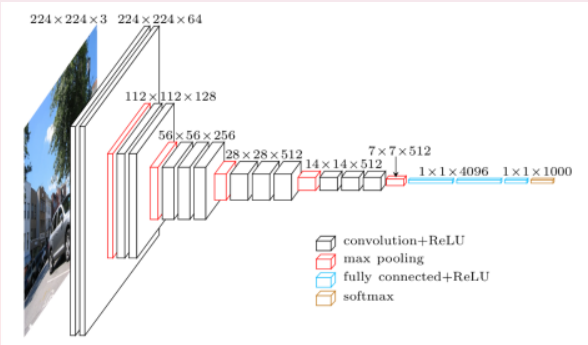
Feb 9, 2018
IPAM workshp on deep learning

Joint work with Qiang Qiu, Jiaji Huang, Robert Calderbank, Guillermo Sapiro, and Ingrid Daubechies

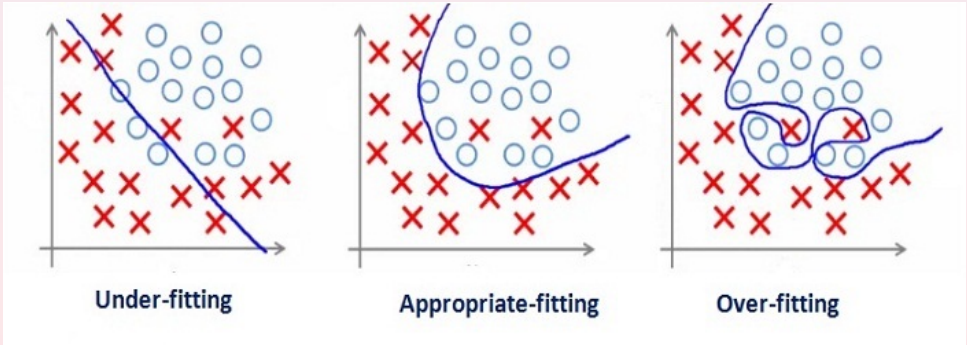
Neural Networks and Overfitting

Deep neural networks (DNNs) have achieved great success in machine learning research and commercial applications. When large amounts of training data are available, the capacity of DNNs can easily be increased by adding more units or layers to extract more effective high-level features.

However, big networks with millions of parameters can easily overfit even the largest of datasets. As a result, the learned network will have low error rates on the training data, but generalizes poorly onto the test data.



Deep Neural Network



Overfitting

Regularizations

Many widely-used network regularizations are data-independent.

- Weight decay
- Parameter sharing
- DropOut
- DropConnect
- Early stopping
- .....

Most of the data-dependent regularizations are motivated by the empirical observation that data of interest typically lie close to a manifold.

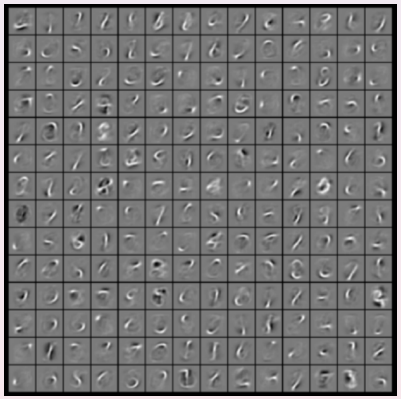
- Tangent distance algorithm
- Tangent prop algorithm
- Manifold tangent classifier
- .....

LDMNet: Low Dimensional Manifold Regularized Neural Networks

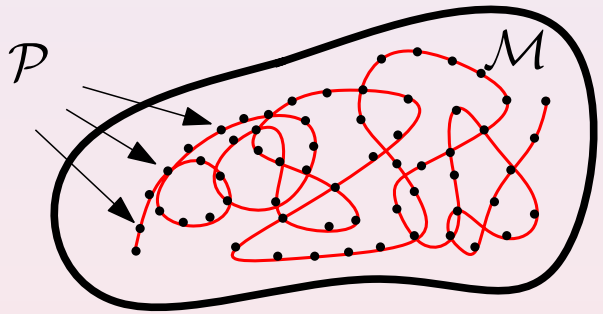
Low Dimensional Manifold Regularized Neural Networks (*LDMNet*) incorporates a feature regularization method that focuses on the geometry of both the input data and the output features.

Input data x_i

Output features ξ_i



Manifold $\mathcal{M}$

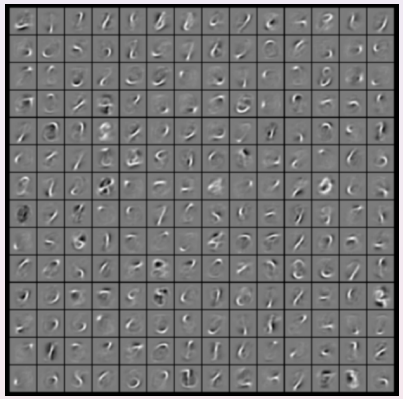


- $\{x_i\}_{i=1}^N \subset \mathbb{R}^{d_1}$ are the input data. $\{\xi_i = f_\theta(x_i)\}_{i=1}^N \subset \mathbb{R}^{d_2}$ are the output features.
- $\mathcal{P} = \{(x_i, \xi_i)\}_{i=1}^N \subset \mathbb{R}^d$ is the collection of the data-feature concatenation (x_i, ξ_i).
- $\mathcal{M} \subset \mathbb{R}^d$ is the underlying manifold, discretely sampled by the point cloud $\mathcal{P}$.

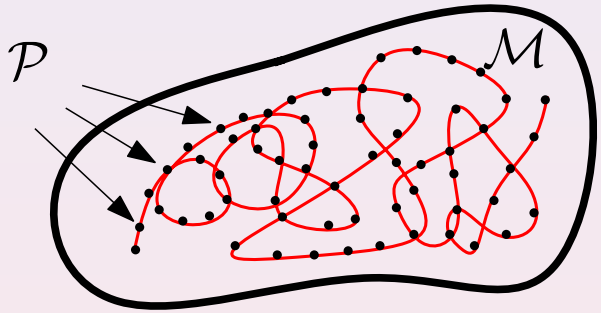
LDMNet: Low Dimensional Manifold Regularized Neural Networks

Input data $\boldsymbol{x}_i$

Output features $\boldsymbol{\xi}_i$



Manifold $\mathcal{M}$



- The input data $\{\boldsymbol{x}_i\}_{i=1}^N$ typically lie close to a collection of low dimensional manifolds, i.e. $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathcal{N} = \cup_{l=1}^L \mathcal{N}_l \subset \mathbb{R}^{d_1}$.
- The feature extractor, $f_{\boldsymbol{\theta}}$, of a good learning algorithm should be a smooth function over $\mathcal{N}$.
- Therefore the concatenation of the input data and output features, $\mathcal{P} = \{(\boldsymbol{x}_i, \boldsymbol{\xi}_i)\}_{i=1}^N$, should sample a collection of low dimensional manifolds $\mathcal{M} = \cup_{l=1}^L \mathcal{M}_l \subset \mathbb{R}^d$, where $d = d_1 + d_2$, and $\mathcal{M}_l = \{(\boldsymbol{x}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))\}_{x \in \mathcal{N}_l}$ is the graph of $f_{\boldsymbol{\theta}}$ over $\mathcal{N}_l$.

LDMNet: Low Dimensional Manifold Regularized Neural Networks

We suggest that network overfitting occurs when $\dim(\mathcal{M}_l)$ is too large after training. Therefore, to reduce overfitting, we explicitly use the dimensions of $\mathcal{M}_l$ as a regularizer in the following variational form:

$$\begin{aligned} \min_{\boldsymbol{\theta},\mathcal{M}} \quad & J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}|}\int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p} \\ \text{s.t.} \quad & \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}, \end{aligned}$$

where for any $\boldsymbol{p} \in \mathcal{M} = \cup_{l=1}^L \mathcal{M}_l$, $\mathcal{M}(\boldsymbol{p})$ denotes the manifold $\mathcal{M}_l$ to which $\boldsymbol{p}$ belongs, and $|\mathcal{M}| = \sum_{l=1}^L |\mathcal{M}_l|$ is the volume of $\mathcal{M}$.

Feature of the MNIST Dataset

10,000 test data

Weight decay

DropOut

LDMNet

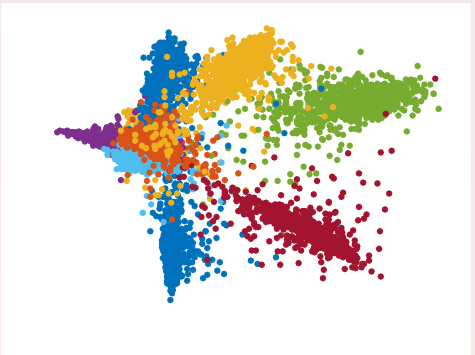


Figure: Test data of MNIST and their features learned by the same network with different regularizers. All networks are trained from the same set of 1,000 images. Data are visualized in two dimensions using PCA, and ten classes are distinguished by different colors.

Dimension of a Manifold

Question: How do we calculate $\dim(\mathcal{M}(\boldsymbol{p}))$ from the point cloud $\mathcal{P} = \{(\boldsymbol{x}_i, f_\theta(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}$

Theorem

Let $\mathcal{M}$ be a smooth submanifold isometrically embedded in $\mathbb{R}^d$. For any $\boldsymbol{p} = (p_i)_{i=1}^d \in \mathcal{M}$,

$$\dim(\mathcal{M}) = \sum_{i=1}^{d} \left|\nabla_{\mathcal{M}} \alpha_i(\boldsymbol{p})\right|^2,$$

where $\alpha_i(\boldsymbol{p}) = p_i$ is the coordinate function, and $\nabla_{\mathcal{M}}$ is the gradient operator on the manifold $\mathcal{M}$. More specifically, $\nabla_{\mathcal{M}} \alpha_i = \sum_{s,t=1}^{k} g^{st} \partial_t \alpha_i \partial_s$, where k is the intrinsic dimension of $\mathcal{M}$, and g^{st} is the inverse of the metric tensor.

Dimension of a Manifold

Sanity check:

If $\mathcal{M} = S^1$, then $k = \dim(\mathcal{M}) = 1$, $d = \dim(\mathbb{R}^2) = 2$, and $\boldsymbol{x} = \psi(\theta) = (\cos\theta, \sin\theta)^t$ is the coordinate chart.
The metric tensor $g = g_{11} = \left\langle \frac{\partial \psi}{\partial \theta}, \frac{\partial \psi}{\partial \theta} \right\rangle = 1 = g^{11}$.
The gradient of α_i, $\nabla_{\mathcal{M}} \alpha_i = g^{11} \partial_1 \alpha_i \partial_1 = \partial_1 \alpha_i \partial_1$ can be viewed as a vector in the ambient space $\mathbb{R}^2$:

$$\nabla^j_{\mathcal{M}} \alpha_i = \partial_1 \psi^j \partial_1 \alpha_i$$

Therefore, we have

$$\nabla_{\mathcal{M}} \alpha_1 = \left\langle \partial_1 \psi^1 \partial_1 \alpha_1, \partial_1 \psi^2 \partial_1 \alpha_1 \right\rangle = \left\langle \sin^2\theta, -\cos\theta \sin\theta \right\rangle,$$
$$\nabla_{\mathcal{M}} \alpha_2 = \left\langle \partial_1 \psi^1 \partial_1 \alpha_2, \partial_1 \psi^2 \partial_1 \alpha_2 \right\rangle = \left\langle -\sin\theta \cos\theta, \cos^2\theta \right\rangle.$$

Hence $\|\nabla_{\mathcal{M}} \alpha_1\|^2 + \|\nabla_{\mathcal{M}} \alpha_2\|^2 = \sin^2\theta + \cos^2\theta = 1$

LDMNet: Low Dimensional Manifold Regularized Neural Networks

$$\begin{aligned} \min_{\boldsymbol{\theta},\mathcal{M}} \quad & J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}|}\int_{\mathcal{M}} \dim(\mathcal{M}(\boldsymbol{p}))d\boldsymbol{p} \\ \text{s.t.} \quad & \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}, \end{aligned} \tag{1}$$

Using Theorem 1, the above variational functional can be reformulated as

$$\begin{aligned} \min_{\boldsymbol{\theta},\mathcal{M}} \quad & J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}|}\sum_{j=1}^{d} \|\nabla_{\mathcal{M}}\alpha_j\|^2_{L^2(\mathcal{M})} \\ \text{s.t.} \quad & \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M} \end{aligned} \tag{2}$$

where $\sum_{j=1}^{d} \|\nabla_{\mathcal{M}}\alpha_j\|^2_{L^2(\mathcal{M})}$ corresponds to the L^1 norm of the local dimension.

How do we solve (2)? Alternate direction of minimization with respect to $\mathcal{M}$ and $\boldsymbol{\theta}$.

Alternate Direction of Minimization

$$\min_{\boldsymbol{\theta},\mathcal{M}} \quad J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}|}\sum_{j=1}^{d}\|\nabla_{\mathcal{M}}\alpha_j\|^2_{L^2(\mathcal{M})}$$

$$\text{s.t.} \quad \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}$$

Given $(\boldsymbol{\theta}^{(k)}, \mathcal{M}^{(k)})$ at step k satisfying $\{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}^{(k)}$, step $k+1$ consists of the following

- Update $\boldsymbol{\theta}^{(k+1)}$ and the perturbed coordinate functions $\boldsymbol{\alpha}^{(k+1)} = (\alpha_1^{(k+1)}, \cdots, \alpha_d^{(k+1)})$ as the minimizers of (3) with the fixed $\mathcal{M}^{(k)}$:

$$\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \quad J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}^{(k)}|}\sum_{j=1}^{d}\|\nabla_{\mathcal{M}^{(k)}}\alpha_j\|^2_{L^2(\mathcal{M}^{(k)})} \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{\alpha}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) = (\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)), \quad \forall i = 1, \ldots, N$$

- Update $\mathcal{M}^{(k+1)}$:

$$\mathcal{M}^{(k+1)} = \boldsymbol{\alpha}^{(k+1)}(\mathcal{M}^{(k)}) \tag{4}$$

Alternate Direction of Minimization

- Update $\boldsymbol{\theta}^{(k+1)}$ and $\boldsymbol{\alpha}^{(k+1)} = (\alpha_1^{(k+1)}, \cdots, \alpha_d^{(k+1)})$ with the fixed $\mathcal{M}^{(k)}$:

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \quad & J(\boldsymbol{\theta}) + \frac{\lambda}{|\mathcal{M}^{(k)}|} \sum_{j=1}^{d} \|\nabla_{\mathcal{M}^{(k)}} \alpha_j\|^2_{L^2(\mathcal{M}^{(k)})} \\ \text{s.t.} \quad & \boldsymbol{\alpha}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) = (\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)), \quad \forall i = 1, \ldots, N \end{aligned}$$

- Update $\mathcal{M}^{(k+1)}$:

$$\mathcal{M}^{(k+1)} = \boldsymbol{\alpha}^{(k+1)}(\mathcal{M}^{(k)})$$

The manifold update is trivial to implement, and the update of $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ is an optimization problem with nonlinear constraint, which "can" be solved via the alternating direction method of multipliers (ADMM).

Alternate Direction of Minimization

$$\min_{\theta,\alpha} \quad J(\theta) + \frac{\lambda}{|\mathcal{M}^{(k)}|}\sum_{j=1}^{d}\|\nabla_{\mathcal{M}^{(k)}}\alpha_j\|^2_{L^2(\mathcal{M}^{(k)})}$$

$$\text{s.t.} \quad \alpha(x_i, f_{\theta^{(k)}}(x_i)) = (x_i, f_\theta(x_i)), \quad \forall i = 1,\dots,N$$

Solve the variationl problem using ADMM. More specifically,

$$\alpha_\xi^{(k+1)} = \arg\min_{\alpha_\xi} \sum_{j=d_1+1}^{d}\|\nabla_{\mathcal{M}^{(k)}}\alpha_j\|_{L^2(\mathcal{M}^{(k)})} + \frac{\mu|\mathcal{M}^{(k)}|}{2\lambda N}\sum_{i=1}^{N}\|\alpha_\xi(x_i, f_{\theta^{(k)}}(x_i)) - (f_{\theta^{(k)}}(x_i) - Z_i^{(k)})\|_2^2.$$

$$\theta^{(k+1)} = \arg\min_\theta J(\theta) + \frac{\mu}{2N}\sum_{i=1}^{N}\|\alpha_\xi^{(k+1)}(x_i, f_{\theta^{(k)}}(x_i)) - (f_\theta(x_i) - Z_i^{(k)})\|_2^2.$$

$$Z_i^{(k+1)} = Z_i^{(k)} + \alpha_\xi^{(k+1)}(x_i, f_{\theta^{(k)}}(x_i)) - f_{\theta^{(k+1)}}(x_i),$$

where $\alpha = (\alpha_x, \alpha_\xi) = ((\alpha_1,\dots,\alpha_{d_1}),(\alpha_{d_1+1},\dots,\alpha_d))$, and Z_i is the dual variable.

ADMM

$$\boldsymbol{\alpha}_{\boldsymbol{\xi}}^{(k+1)} = \arg\min_{\boldsymbol{\alpha}_{\boldsymbol{\xi}}} \sum_{j=d_1+1}^{d} \|\nabla_{\mathcal{M}^{(k)}} \alpha_j\|_{L^2(\mathcal{M}^{(k)})}$$
$$+ \frac{\mu|\mathcal{M}^{(k)}|}{2\lambda N} \sum_{i=1}^{N} \|\boldsymbol{\alpha}_{\boldsymbol{\xi}}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) - (f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i) - Z_i^{(k)})\|_2^2. \quad (5)$$

$$\boldsymbol{\theta}^{(k+1)} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \frac{\mu}{2N} \sum_{i=1}^{N} \|\boldsymbol{\alpha}_{\boldsymbol{\xi}}^{(k+1)}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) - (f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - Z_i^{(k)})\|_2^2. \quad (6)$$

$$Z_i^{(k+1)} = Z_i^{(k)} + \boldsymbol{\alpha}_{\boldsymbol{\xi}}^{(k+1)}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) - f_{\boldsymbol{\theta}^{(k+1)}}(\boldsymbol{x}_i), \quad (7)$$

Among (5),(6) and (7), (7) is the easiest to implement, (6) can be solved by stochastic gradient descent (SGD) with modified back propagation, and (5) can be solved by the point integral method (PIM) [Z. Shi, J. Sun].

Shi, Z., and Sun, J. (2013). Convergence of the point integral method for the poisson equation on manifolds ii: the dirichlet boundary. arXiv preprint arXiv:1312.4424

Back Propagation for the θ Update

$$\begin{aligned}\theta^{(k+1)} &= \arg\min_{\theta} J(\theta) + \frac{\mu}{2N}\sum_{i=1}^{N} \|\alpha_{\xi}^{(k+1)}(x_i, f_{\theta^{(k)}}(x_i)) - (f_{\theta}(x_i) - Z_i^{(k)})\|_2^2. \\ &= \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \ell(f_{\theta}(x_i), y_i) + \frac{1}{N}\sum_{i=1}^{N} E_i(\theta), \end{aligned} \tag{8}$$

where $E_i(\theta) = \frac{\mu}{2}\|\alpha_{\xi}^{(k+1)}(x_i, f_{\theta^{(k)}}(x_i)) - (f_{\theta}(x_i) - Z_i^{(k)})\|_2^2$. The gradient of the second term with respect to the output layer $f_{\theta}(x_i)$ is:

$$\frac{\partial E_i}{\partial f_{\theta}(x_i)} = \mu\left(f_{\theta}(x_i) - Z_i^{(k)} - \alpha_{\xi}^{(k+1)}(x_i, f_{\theta^{(k)}}(x_i))\right) \tag{9}$$

This means that we need to only add the extra term (9) to the original gradient, and then use the already known procedure to back-propagate the gradient.

Point Integral Method for the α Update

$$\boldsymbol{\alpha}_{\boldsymbol{\xi}}^{(k+1)} = \arg\min_{\boldsymbol{\alpha}_{\boldsymbol{\xi}}} \sum_{j=d_1+1}^{d} \|\nabla_{\mathcal{M}^{(k)}} \alpha_j\|_{L^2(\mathcal{M}^{(k)})}$$
$$+ \frac{\mu|\mathcal{M}^{(k)}|}{2\lambda N} \sum_{i=1}^{N} \|\boldsymbol{\alpha}_{\boldsymbol{\xi}}(\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)) - (f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i) - Z_i^{(k)})\|_2^2.$$

Note that the objective funtion above is decoupled with respect to j, and each α_j update can be cast into:

$$\min_{u \in H^1(\mathcal{M})} \|\nabla_{\mathcal{M}} u\|^2_{L^2(\mathcal{M})} + \gamma \sum_{\boldsymbol{q} \in P} |u(\boldsymbol{q}) - v(\boldsymbol{q})|^2, \tag{10}$$

where $u = \alpha_j$, $\mathcal{M} = \mathcal{M}^{(k)}$,$P = \{\boldsymbol{p}_i = (\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}$, and $\gamma = \mu|\mathcal{M}^{(k)}|/2\lambda N$. The Euler-Lagrange equation of (10) is:

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q} \in P} \delta(\boldsymbol{p} - \boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, \ \boldsymbol{p} \in \mathcal{M} \\ \\ \dfrac{\partial u}{\partial n} = 0, \ \boldsymbol{p} \in \partial\mathcal{M} \end{cases}$$

Point Integral Method

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q} \in P} \delta(\boldsymbol{p} - \boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, \ \boldsymbol{p} \in \mathcal{M} \\ \dfrac{\partial u}{\partial n} = 0, \ \boldsymbol{p} \in \partial \mathcal{M} \end{cases}$$

In the point integral method (PIM) [Z. Shi, J. Sun], the key observation is the following integral approximation:

$$\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y}) \bar{R}\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right) d\boldsymbol{y} \approx -\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\, R\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right) d\boldsymbol{y} + 2\int_{\partial \mathcal{M}} \frac{\partial u}{\partial n}(\boldsymbol{y}) \bar{R}\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right) d\tau_{\boldsymbol{y}}.$$

The function R is a positive function defined on $[0, +\infty)$ with compact support (or fast decay) and

$$\bar{R} = \int_r^{\infty} R(s) ds.$$

Local Truncation Error

Theorem

Let $\mathcal{M}$ be a smooth manifold and $u \in C^3(\mathcal{M})$, then

$$\left\| -\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\, R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} + 2\int_{\partial\mathcal{M}} \frac{\partial u}{\partial n}(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) d\tau_{\boldsymbol{y}} - \int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} \right\|_{L^2(\mathcal{M})} = O(t^{1/4}),$$

where

$$R_t(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{(4\pi t)^{k/2}} R\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right), \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{(4\pi t)^{k/2}} \bar{R}\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right).$$

Point Integral Method for the α Update

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q} \in P} \delta(\boldsymbol{p} - \boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, \ \boldsymbol{p} \in \mathcal{M} \\ \dfrac{\partial u}{\partial n} = 0, \ \boldsymbol{p} \in \partial \mathcal{M} \end{cases}$$

After convolving the above equation with the heat kernel $R_t(\boldsymbol{p}, \boldsymbol{q}) = C_t \exp\left(-\frac{|\boldsymbol{p}-\boldsymbol{q}|^2}{4t}\right)$, we know the solution u should satisfy

$$-\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{q}) R_t(\boldsymbol{p}, \boldsymbol{q}) d\boldsymbol{q} + \gamma \sum_{\boldsymbol{q} \in P} R_t(\boldsymbol{p}, \boldsymbol{q}) \left(u(\boldsymbol{q}) - v(\boldsymbol{q})\right) = 0. \tag{11}$$

Combined with Theorem 2 and the Neumann boundary condition, this implies that u should approximately satisfy

$$\int_{\mathcal{M}} \left(u(\boldsymbol{p}) - u(\boldsymbol{q})\right) R_t(\boldsymbol{p}, \boldsymbol{q}) d\boldsymbol{q} + \gamma t \sum_{\boldsymbol{q} \in P} R_t(\boldsymbol{p}, \boldsymbol{q}) \left(u(\boldsymbol{q}) - v(\boldsymbol{q})\right) = 0 \tag{12}$$

Point Integral Method for the α Update

$$\int_{\mathcal{M}} (u(\boldsymbol{p}) - u(\boldsymbol{q}))\, R_t(\boldsymbol{p}, \boldsymbol{q}) d\boldsymbol{q} + \gamma t \sum_{\boldsymbol{q} \in P} R_t(\boldsymbol{p}, \boldsymbol{q})\, (u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0$$

Assume that $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$ samples the manifold $\mathcal{M}$ uniformly at random, then the above equation can be discretized as

$$\frac{|\mathcal{M}|}{N} \sum_{j=1}^{N} R_{t,ij}(u_i - u_j) + \gamma t \sum_{j=1}^{N} R_{t,ij}(u_j - v_j) = 0, \tag{13}$$

where $u_i = u(\boldsymbol{p}_i)$, and $R_{t,ij} = R_t(\boldsymbol{p}_i, \boldsymbol{p}_j)$. Combining the definition of γ in (10), we can write (13) in the matrix form

$$\left(\boldsymbol{L} + \frac{\mu}{\tilde{\lambda}} \boldsymbol{W}\right) \boldsymbol{u} = \frac{\mu}{\tilde{\lambda}} \boldsymbol{W} \boldsymbol{v}, \quad \tilde{\lambda} = 2\lambda/t, \tag{14}$$

where $\tilde{\lambda}$ can be chosen instead of λ as the hyperparameter to be tuned, $\boldsymbol{u} = (u_1, \ldots, u_N)^T$, $\boldsymbol{W}$ is an $N \times N$ matrix

$$\boldsymbol{W}_{ij} = R_{t,ij} = \exp\left(-\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|^2}{4t}\right), \tag{15}$$

and $\boldsymbol{L}$ is the graph Laplacian of $\boldsymbol{W}$:

Algorithm for Training *LDMNet*

Algorithm 1 LDMNet Training

Input: Training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \subset \mathbb{R}^{d_1} \times \mathbb{R}$, hyperparameters $\tilde{\lambda}$ and μ, and a neural network with the weights $\boldsymbol{\theta}$ and the output layer $\boldsymbol{\xi}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \in \mathbb{R}^{d_2}$.

Output: Trained network weights $\boldsymbol{\theta}^*$.

1: Randomly initialize the network weights $\boldsymbol{\theta}^{(0)}$. The dual variables $Z_i^{(0)} \in \mathbb{R}^{d_2}$ are initialized to zero.
2: **while** not converge **do**
3: 1. Compute the matrices $\boldsymbol{W}$ and $\boldsymbol{L}$ as in (15) and (16) with $\boldsymbol{p}_i = (\boldsymbol{x}_i, f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i))$.
4: 2. Update $\boldsymbol{\alpha}^{(k+1)}$ in (5): solve the linear systems (14), where
$$\boldsymbol{u}_i = \alpha_j(\boldsymbol{p}_i), \quad \boldsymbol{v}_i = f_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_i)_j - Z_{i,j}^{(k)}.$$
5: 3. Update $\boldsymbol{\theta}^{(k+1)}$ in (6): run SGD for M epochs with an extra gradient term (9).
6: 4. Update $Z^{(k+1)}$ in (7).
7: 5. $k \leftarrow k + 1$.
8: **end while**
9: $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^{(k)}$.

Complexity Analysis

The additional computation in Algorithm 1 comes from the update of weight matrices and solving the linear system from PIM once every M epochs of SGD.

The weight matrix $\boldsymbol{W}$ is truncated to only 20 nearest neighbors. To identify those nearest neighbors, we first organize the data points $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\} \subset \mathbb{R}^d$ into a k-d tree. Nearest neighbors can then be efficiently identified because branches can be eliminated from the search space quickly. Modern algorithms to build a balanced k-d tree generally at worst converge in $O(N \log N)$ time, and finding nearest neighbours for one query point in a balanced k-d tree takes $O(\log N)$ time on average. Therefore the complexity of the weight update is $O(N \log N)$.

Since $\boldsymbol{W}$ and $\boldsymbol{L}$ are sparse symmetric matrices with a fixed maximum number of non-zero entries in each row, the linear system can be solved efficiently with the preconditioned conjugate gradients method. After restricting the number of matrix multiplications to a maximum of 50, the complexity of the α update is $O(N)$.

Experimental Setup

We compare the performance of *LDMNet* to widely-used network regularization techniques, weight decay and *DropOut*, using the same underlying network structure.

Unless otherwise stated, all experiments use mini-batch SGD with momentum on batches of 100 images. The momentum parameter is fixed at 0.9. The networks are trained using a fixed learning rate r_0 on the first 200 epochs, and then $r_0/10$ for another 100 epochs.

For *LDMNet*, the weight matrices and α are updated once every $M = 2$ epochs of SGD. For *DropOut*, the corresponding *DropOut* layer is always chosen to have a drop rate of 0.5. All other network hyperparameters are chosen according to the error rate on the validation set.

MNIST

The MNIST handwritten digit dataset contains approximately 60,000 training images (28×28) and 10,000 test images.

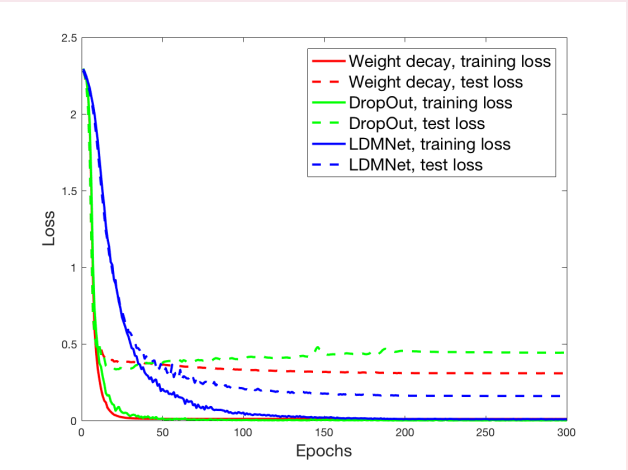
Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 1 \times 20$ stride: 1, pad: 0
2	max pool	size: 2×2, stride: 2, pad: 0
3	conv	size: $5 \times 5 \times 20 \times 50$ stride: 1, pad: 0
4	max pool	size: 2×2, stride: 2, pad: 0
5	conv	size: $4 \times 4 \times 50 \times 500$ stride: 1, pad: 0
6	ReLu (*DropOut*)	N/A
7	fully connected	500×10
8	softmaxloss	N/A

Table: Network structure in the MNIST experiments. The outputs of layer 6 are the extracted features, which will be fed into the softmax classifier (layer 7 and 8).

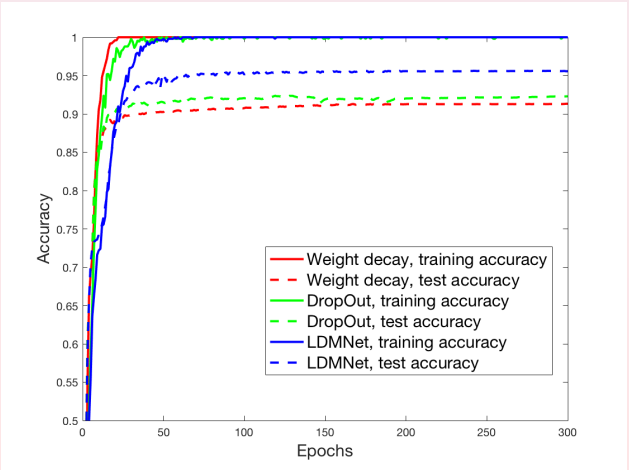
MNIST

training per class	weight decay	*DropOut*	*LDMNet*
50	91.32%	92.31%	**95.57%**
100	93.38%	94.05%	**96.73%**
400	97.23%	97.95%	**98.41%**
700	97.67%	98.07%	**98.61%**
1000	98.06%	98.71%	**98.89%**
3000	98.87%	99.21%	**99.24%**
6000	99.15%	**99.41%**	99.39%

Loss



Accuracy



MNIST

10,000 test data

Weight decay

DropOut

LDMNet

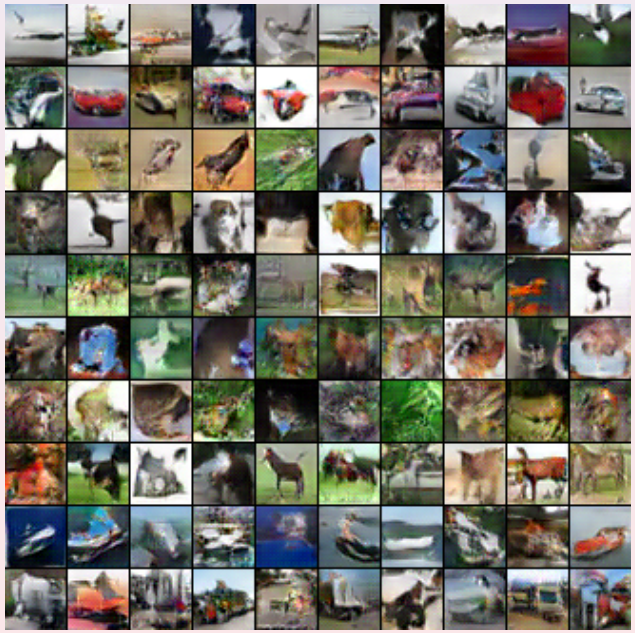
Figure: Test data of MNIST and their features learned by the same network with different regularizers. All networks are trained from the same set of 1,000 images. Data are visualized in two dimensions using PCA, and ten classes are distinguished by different colors.

SVHN and CIFAR-10

SVHN and CIFAR-10 are benchmark RGB image datasets, both of which contain 10 different classes.

SVHN

CIFAR-10



SVHN and CIFAR 10

Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 3 \times 96$ stride: 1, pad: 2
2	ReLu	N/A
3	max pool	size: 3×3, stride: 2, pad: 0
4	conv	size: $5 \times 5 \times 96 \times 128$ stride: 1, pad: 2
5	ReLu	N/A
6	max pool	size: 3×3, stride: 2, pad: 0
7	conv	size: $4 \times 4 \times 128 \times 256$ stride: 1, pad: 0
8	ReLu	N/A
9	max pool	size: 3×3, stride: 2, pad: 0
10	fully connected	output: 2048
11	ReLu (*DropOut*)	N/A
12	fully connected	output: 2048
13	ReLu (*DropOut*)	N/A
14	fully connected	2048×10
15	softmaxloss	N/A

SVHN and CIFAR 10

training per class	weight decay	*DropOut*	*LDMNet*
50	71.46%	71.94%	**74.64%**
100	79.05%	79.94%	**81.36%**
400	87.38%	87.16%	**88.03%**
700	89.69%	89.83%	**90.07%**

Table: SVHN: testing accuracy for different regularizers

training per class	weight decay	*DropOut*	*LDMNet*
50	34.70%	35.94%	**41.55%**
100	42.45%	43.18%	**48.73%**
400	56.19%	56.79%	**60.08%**
700	61.84%	62.59%	**65.59%**
full data	87.72%		**88.21%**

Table: CIFAR-10: testing accuracy for different regularizers.

NIR-VIS Heterogeneous Face Recognition

The objective of the experiment is to match a probe image of a subject captured in the near-infrared spectrum (NIR) to the same subject from a gallery of visible spectrum (VIS) images. The CASIA NIR-VIS 2.0 benchmark dataset is used to evaluate the performance.

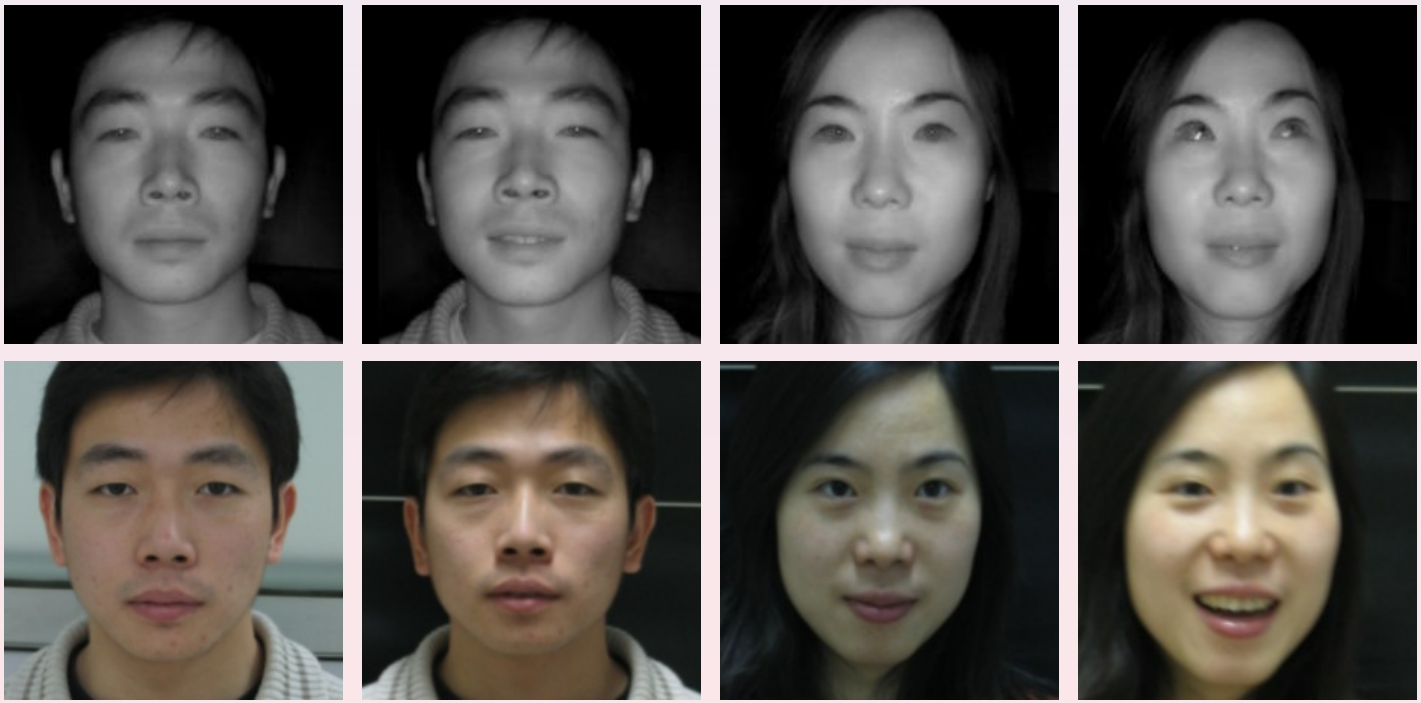


Figure: Sample images of two subjects from the CASIA NIR-VIS 2.0 dataset after the pre-procssing of alignment and cropping. Top: NIR. Bottom: VIS.

NIR-VIS Heterogeneous Face Recognition

Despite recent breakthroughs for VIS face recognition by training DNNs from millions of VIS images, such approach cannot be simply transferred to NIR-VIS face recognition.

- Unlike VIS face images, we have only limited number of availabe NIR images.
- The NIR-VIS face matching is a cross-modality comparison.

The authors in [Lezama et al., 2017] introduced a way to transfer the breakthrough in VIS face recognition to the NIR spectrum. Their idea is to use a DNN pre-trained on VIS images as a feature extactor, while making two independent modifications in the input and output of the DNN.

- Modify the input by "hallucinating" a VIS image from the NIR sample.
- Learn an embedding of the DNN features at the output

Lezama, J., Qiu, Q., and Sapiro, G. (2017). Not afraid of the dark: Nir-vis face recognition via cross-spectral hallucination and low-rank embedding. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

NIR-VIS Heterogeneous Face Recognition

- Modify the input by "hallucinating" a VIS image from the NIR sample.
- Learn an embedding of the DNN features at the output

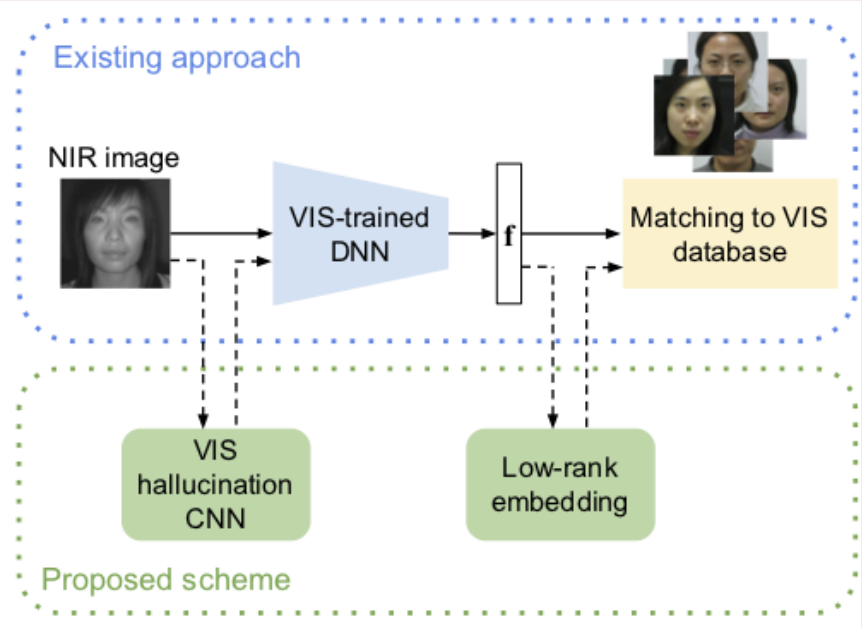


Figure: Proposed procedure in [Lezama et al., 2017]

NIR-VIS Heterogeneous Face Recognition

We follow the second idea in [Lezama et al., 2017], and learn a nonlinear low dimensional manifold embedding of the output features. we use the VGG-face model as a feature extractor. We then put the 4,096 dimensional features into a two-layer fully connected network to learn a nonlinear embedding using different regularizations.

Layer	Type	Parameters
1	fully connected	output:2000
2	ReLu (*DropOut*)	N/A
3	fully connected	output:2000
4	ReLu (*DropOut*)	N/A

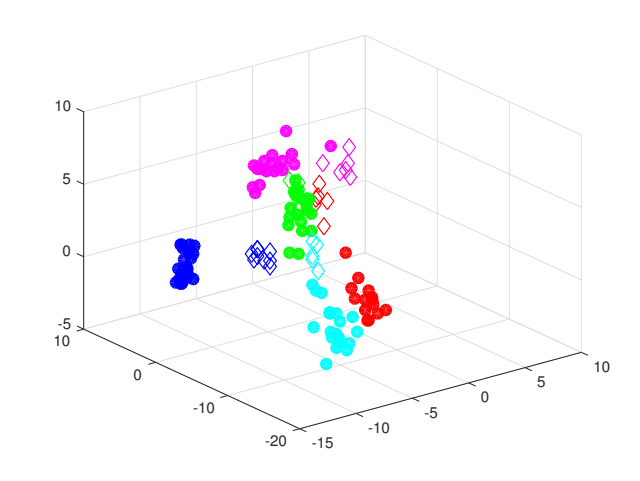
Table: Fully connected network for the NIR-VIS nonlinear feature embedding. The outputs of layer 4 are the extracted features.

NIR-VIS Heterogeneous Face Recognition

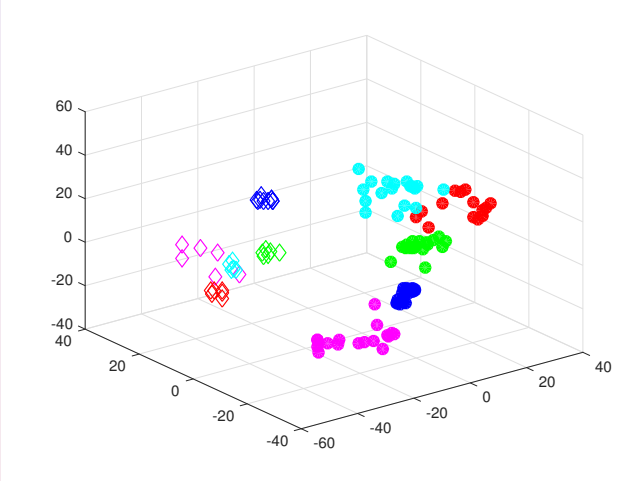
	Accuracy (%)
VGG-face	74.51 ± 1.28
VGG-face + triplet [Lezama et al., 2017]	75.96 ± 2.90
VGG-face + low-rank [Lezama et al., 2017]	80.69 ± 1.02
VGG-face weight Decay	63.87 ± 1.33
VGG-face *DropOut*	66.97 ± 1.31
VGG-face *LDMNet*	$\mathbf{85.02 \pm 0.86}$

NIR-VIS Heterogeneous Face Recognition

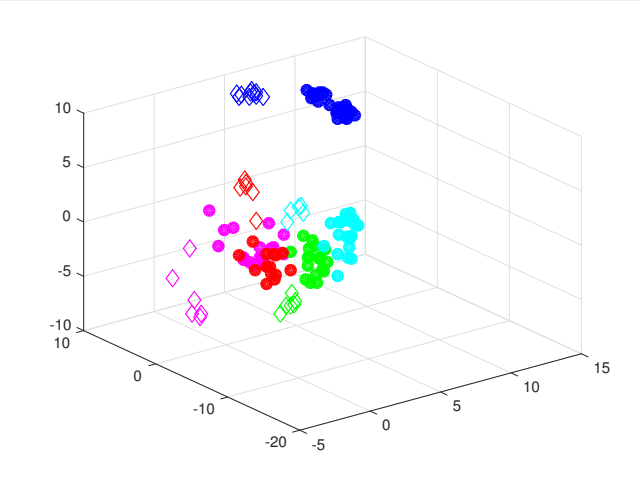
VGG-face



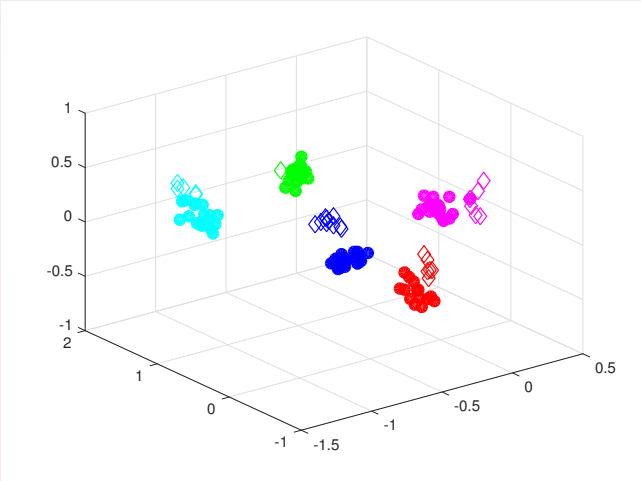
Weight decay



DropOut



LDMNet



Conclusion

- *LDMNet* is a general network regularization technique that focuses on the geometry of both the input data and the output feature
- *LDMNet* directly minimize the dimension of the manifolds without explicit parametrization.
- *LDMNet* significantly outperforms the widely-used network regularizers.
- Limits: data augmentation, $O(N \log N)$ complexity.

Thank you