Nearest Neighbour Radial Basis Function Solvers for Deep Neural Networks

Benjamin J. Meyer, Ben Harwood, Tom Drummond
ARC Centre of Excellence for Robotic Vision
Monash University
{benjamin.meyer,ben.harwood,tom.drummond}@monash.edu

Abstract

We present a radial basis function solver for convolutional neural networks that can be directly applied to both image classification and distance metric learning problems. Our method treats all training features from a deep neural network as radial basis function centres and computes loss by summing the influence of a feature’s nearby centres in the embedding space. Having a radial basis function centred on each training feature is made scalable by treating it as an approximate nearest neighbour search problem. End-to-end learning of the network and solver is carried out, mapping high dimensional features into clusters of the same class. This results in a well-formed embedding space, where semantically related instances are likely to be located near one another, regardless of whether or not the network was trained on those classes. The same loss function is used for both the metric learning and classification problems. We show that our radial basis function solver sets state-of-the-art embedding results on the Stanford Cars196 and CUB-200-2011 datasets. Additionally, we show that when used as a classifier, our method outperforms a conventional softmax classifier on the Caltech-256 object recognition dataset and the fine-grained recognition dataset CUB-200-2011.

1 Introduction

The solver of a neural network is vital to its performance, as it defines the objective and drives the learning. We define a solver as the layers of the network that are aware of the class labels of the data. In the domain of image classification, a softmax solver is conventionally used to transform activations into a distribution across class labels [11, 18, 23, 8]. While in the domain of distance metric learning, a Siamese [3] or triplet [2, 17, 13] solver, with contrastive or hinge loss, is commonly used to pull embeddings of the same class together and push embeddings of different classes apart. The two tasks of classification and metric learning are related but distinct. Conventional classification learning is generally used when the objective is to associate data with a pre-defined set of classes and there is sufficient data to train or fine-tune a network to do so. Distance metric learning, or embedding space building, aims to learn an embedding space where samples with similar semantic meaning are located near one another. Applications for learning such effective embeddings include transfer learning, retrieval, clustering and weakly supervised or self-supervised learning.

In this paper, we present a deep neural network solver that can be applied to both embedding space building and classification problems. The solver defines training features in the embedding space as radial basis function (RBF) centres, which are used to push or pull features in a local neighbourhood, depending on the labels of the associated training samples. The same loss function is used for both classification and metric learning problems. This means that a network trained for the classification task results in feature embeddings of the same class being located near one another and similarly, a network trained for metric learning results in feature embeddings that can be well classified by our
The best success on embedding building tasks has been achieved by deep metric learning methods [9, 17, 21, 19, 13], which make use of deep neural networks. Such approaches may indiscriminately pull samples of the same class together, regardless of whether these two samples were already within well defined local clusters of like samples. These methods aim to form a single cluster per class. In contrast, our approach pushes a feature around the embedding space based only on the local neighbourhood of that feature. This means that the current structure of the space is considered, allowing multiple clusters to form for a single class, if that is appropriate. Our radial basis function solver is able to learn embeddings that result in samples of similar semantic meaning being located near one another. Our experiments show that the RBF solver is able to do this better than existing deep metric learning methods.

Softmax solvers have been a mainstay of the standard classification problem [11, 18, 23, 8]. Such an approach is inefficient as classes must be axis-aligned and the number of classes is baked into the network. In our RBF approach, classes do not need to be axis-aligned and as such multiple clusters may form for a single class. The nearest neighbour RBF solver outperforms conventional softmax solvers in our experiments and provides additional adaptability and flexibility, as new classes can be added to the problem with no updates to the network weights required to obtain reasonable results. This performance improvement is obtained despite smaller model capacity. The RBF solver by its very nature is a classifier, but learns the classification problem in the exact same way it learns the embedding space building problem.

The main advantages of our novel radial basis function solver for neural networks can be summarised as follows:

- Our solver can be directly applied to two previously separate problems; classification and embedding space learning.
- End-to-end learning can be carried out in an efficient manner by leveraging fast approximate nearest neighbour search (as seen in Section 3.2).
- Our approach outperforms current state-of-the-art deep metric learning algorithms on the Stanford Cars196 and CUB-200-2011 datasets (as seen in Section 4.1).
- Finally, our radial basis function classifier outperforms a conventional softmax classifier on the object recognition dataset Caltech-256 and the fine-grained recognition dataset CUB-200-2011 (as seen in Section 4.2).

2 Related Work

Radial Basis Functions in Neural Networks  
Radial basis function networks were introduced by Broomhead and Lowe [4]. The networks formulate activation functions as RBFs, resulting in an output that is a sum of radial basis function values between the input and network parameters. In contrast to these radial basis function networks, our approach uses RBFs in the solver of a deep convolutional neural network and our radial basis function centres are coupled to high dimensional embeddings of training samples, rather than being network parameters. Radial basis functions have been used as neural network solvers in the form of support vector machines. In one such formulation, a neural network is used as a fixed feature extractor and separate support vector machines are trained to classify the features [13, 4]. No joint training occurs between the solver (classifier) and network. Such an approach is often used for transfer learning, where the network is trained on vast amounts of data and the support vector machines are trained for problems in which labelled training data is scarce. Tang [24] replaces the typical softmax classifier with linear support vector machines. In this case, the solver and network are trained jointly, meaning the loss that is minimised is margin based.

Metric Learning  
Early methods in the domain of metric learning include those that use Siamese networks [11] and contrastive loss [6, 3]. The objective of these approaches is to pull pairwise samples of the same class together and push pairwise samples of different classes apart. Such methods work on absolute distances, while triplet networks with hinge loss [27] work on relative distance. Triplet loss approaches take a trio of inputs; an anchor, a positive sample of the same class as the anchor and a negative sample of a different class. Triplet loss aims to pull the positive sample closer to the anchor and
than the negative sample. Several deep metric learning approaches make use of, or generalise deep triplet neural networks [9, 26, 17, 21, 19, 13]. Schroff et al. [17] perform semi-hard mining within a mini-batch, while Song et al. [21] propose a lifted structured embedding with efficient computation of the full distance matrix within a mini-batch. This allows comparisons between all positive and negative pairs in the batch. Similarly, Sohn [19] proposes an approach that allows multiple intra-batch distance comparisons, but optimises a generalisation of triplet loss, named N-pair loss, rather than a max-margin based objective, as in [21]. The global embedding structure is considered in [20] by directly minimising a global clustering metric, while a combination of global and triplet loss is shown to be beneficial in [12]. Finally, Kumar et al. [13] introduce a smart mining technique that mines for triplets over the entire dataset. A Fast Approximate Nearest Neighbour Graph (FANNG) [7] is leveraged for computational efficiency.

### 3 Radial Basis Function Solvers

A radial basis function returns a value that depends only on the distance between two points, one of which is commonly referred to as a centre. Although several radial basis functions exist, in this paper we use RBF to refer to a Gaussian radial basis function, which returns a value based on the Euclidean distance between a point $x$ and the RBF centre $c$. The radial basis function, $f$, is calculated as:

$$f(x, c) = \exp\left(-\frac{\|x - c\|^2}{2\sigma^2}\right)$$

(1)

where $\sigma$ is standard deviation that controls the width of the Gaussian curve, that is, the region around the RBF centre deemed to be of importance.

In the context of our neural network solver, we define the deep feature embeddings of each training set sample as radial basis function centres. Specifically, we take the layer in a network immediately before the solver as the embedding layer. For example, in a VGG architecture, this may be FC7 (fully connected layer 7), forming a 4096 dimension embedding. In general, however, the embedding may be of any size. An overview of this approach is seen in Figure 1.

#### 3.1 Classifier and Loss Function

A radial basis function classifier can be formed by the weighted sum of the RBF distance calculations between a sample feature embedding and the centres. Classification of a sample is achieved by passing the input through the network, resulting in a feature embedding in the same space as the RBF centres. A probability distribution over class labels is found by summing the influence of each centre and normalising. A centre contributes only to the probability of the ground truth label of the training sample coupled to that centre. For example, the probability that the feature embedding $x$ has class label $Q$ is:

$$Pr(x \in \text{class } Q) = \frac{\sum_{i \in Q} w_i f(x, c_i)}{\sum_{m=1}^m w_j f(x, c_j)}$$

(2)

where $f$ is the RBF, $i \in Q$ are the centres with label $Q$, $m$ is the number of training samples and $w_i$ is a learnable weight for RBF centre $i$. Of course, if a sample is in the training set and has a
corresponding RBF centre, the distance calculation to itself is omitted during the computation of the classification distribution, the loss function and the derivatives.

The loss function used for optimisation is simply the summed negative logarithm of the probabilities of the true class labels. For example, the loss $L$ for sample $x$ with ground truth label $R$ is:

$$L(x) = -\ln (Pr(x \in \text{class } R)).$$  \hspace{1cm} (3)

The same loss function is used regardless of whether the network is being trained for classification, as above, or for embedding space building (distance metric learning). This is possible since the RBF classifier is directly computed from distances between features in the embedding space. This means that a network trained for classification will result in features of the same class being located near one another, and similarly a network trained for metric learning will result in an embedding space in which features can be well classified using RBFs.

### 3.2 Nearest Neighbour RBF Solver

In Equation [2] the distribution is calculated by summing over all RBF centres. However, since these centres are attached to training samples, of which there could be any large number, computing that sum is both intractable and unnecessary. The majority of RBF values for a given feature embedding will be effectively zero, as the sample feature will lie only within a subset of the RBF centres’ Gaussian windows. As such, only the local neighbourhood around a feature embedding should be considered. Operating on the set of the nearest RBF centres to a feature ensures that most of the distance values computed are pertinent to the loss calculation. The classifier equation becomes:

$$Pr(x \in \text{class } Q) = \frac{\sum_{i \in Q \cap N} w_i f(x, c_i)}{\sum_{j \in N} w_j f(x, c_j)},$$  \hspace{1cm} (4)

where $N$ is the set of approximate nearest neighbours for sample $x$ and therefore $i \in Q \cap N$ is the set of approximate nearest neighbours that have label $Q$. Again, we note that training set samples exclude their own RBF centre from their nearest neighbour list.

We use approximate nearest neighbour search to obtain candidate nearest neighbour lists. This allows for a trade off between precision and computational efficiency. Specifically, we use a Fast Approximate Nearest Neighbour Graph (FANNG) [7], as it provides the most efficiency when needing a high probability of finding the true nearest neighbours of a query point. Importantly, FANNG provides a scalable solution in terms of the number of dimensions and the number of training samples.

### 3.3 End-to-end Learning

The network and solver weights are learnt end-to-end. Since the weights are constantly being updated during training, the RBF centres are moving and consequently the structure of the nearest neighbour graph is changing. For the nearest neighbour graph to remain completely accurate, it would have to be rebuilt every time the network changes, that is, every time a batch is back propagated through the network. In practice, this is intractable and is remedied by simply considering a larger number of nearest neighbours than would be needed if the graph were to be continuously rebuilt. Due to the nature of approximate nearest neighbour search, these additional nearest neighbours come from an expanding search around the target area. The embedding space changes slowly enough that it is highly likely many of the previously neighbouring RBF centres will remain relevant. Since the Gaussian RBF decays to zero as the distance between the points becomes large, it does not matter if an RBF centre that is no longer near the sample remains a candidate nearest neighbour. Although it is necessary to rebuild the graph to obtain optimal performance, considering a large number of nearest neighbours allows us to rebuild the graph at less frequent intervals. We find rebuilding the graph every 5-10 epochs and considering between 100 and 500 nearest neighbours per sample yields good results for the datasets we consider in Section 4.

A related problem arises in the calculation of the derivatives of the loss with respect to the features. This calculation requires dimension by dimension differences between the feature embeddings and the RBF centres. The centres are moving as the network is being updated, but computing the current RBF centre locations online is intractable. For example, if considering 100 nearest neighbours, 101 samples would need to be propagated forward through the network for each training sample. We find that is is not necessary for every RBF centre to be up to date at all times in order for the model
to converge. A bank of the RBF centres is stored and updated at a fixed interval. In practice, we update the RBF centres when the nearest neighbour graph is being rebuilt, as that also requires a full forward pass of the training data. However, other update schemes could be used. Centres can also be updated on the fly during training; forward passing a training batch during learning yields the up to date centres for those samples. Note that the stored RBF centres do not have dropout [22] applied to them, but the current training sample embeddings may, in order to reduce overfitting.

**Radial Basis Function Parameters** A global standard deviation parameter is shared amongst the RBFs. This ensures that the assumption made about samples only being influenced by their nearest RBF centres holds. Although the parameter is learnable during training, our experiments find that the network will converge over a rather large range of fixed standard deviation values and there is no real performance loss or gain between having the standard deviation as a learnable parameter or fixing it at a reasonable value at the commencement of training. As seen in Equation 4, each RBF centre has a weight, which is learnt end-to-end with the network weights. We only learn RBF weights for the classification task; they are fixed at a value of one for embedding space learning problems. Our experiments find that the performance gain from the RBF weights is small.

4 Experiments

We detail our experimental results in two tasks; distance metric learning and image classification.

4.1 Distance Metric Learning

**Experimental Set-up** We evaluate our approach on two datasets; Stanford Cars196 [10] and CUB-200-2011 (Birds200) [28]. Cars196 consists of 16,185 images of 196 different car makes and models, while Birds200 consists of 11,788 images of 200 different bird species. In this problem, the network is trained and evaluated on different sets of classes. We follow the experimental set-up used in [21, 19, 20, 13]. For the Cars196 dataset, we train the network on the first 98 classes and evaluate on the remaining 98. For the Birds200 dataset we train on the first 100 classes and evaluate on the remaining 100. All images are resized to be 256x256 and we do not crop the images using the provided bounding boxes.

Our method is compared to the state-of-the-art approaches on the considered datasets; semi-hard mining for triplet networks [17], lifted structured feature embedding [21], N-pair loss [19], clustering [20], global loss with triplet networks [12] and smart mining for triplet networks [13]. For fair comparison to these methods, we use the same base architecture for our experiments; GoogLeNet [23]. Network weights are initialised from ImageNet [16] pre-trained weights. We use 100 nearest neighbours from FANNG, with the graph rebuilt every 10 epochs. RBF weights are fixed at a value of one for this task. We train for 50 epochs on Cars196 and 30 epochs on Birds200. A base learning of 0.00001 and weight decay of 0.0002 are used.

**Evaluation Metrics** Following [21], we evaluate the embedding space using two metrics; Normalised Mutual Information (NMI) [14] and Recall@K. The NMI score is the ratio of mutual information and average entropy of a set of clusters and labels. It evaluates only for the number of clusters equal to the number of classes. As discussed in Section 1, a good embedding does not necessarily have only one cluster per class, but may have multiple well formed clusters in the space. This means that our mutual information may be higher than reported with this metric. Nevertheless, we present results on the NMI score in the interest of comparing to existing methods that evaluate on this metric. The Recall@K (R@K) metric is better suited for evaluating an embedding space. A true positive is defined as a sample that has at least one of its true nearest K neighbours in the embedding space with the same class as itself.

**Embedding Space Dimension** We investigate the importance of the embedding dimension. A similar study in [21] suggests that the number of dimensions is not important for triplet networks, in fact, increasing the number of dimensions can be detrimental to performance. We compare our method with increasing dimension size against triplet loss [27, 17] and lifted structured embedding [21], both taken from the study in [21]. Figure 2 shows the affect of the embedding size on NMI score. It’s clear that while increasing the number of dimensions does not necessarily improve performance for triplet-based networks, the dimensionality is important for our RBF approach. The NMI score for
Comparison of Results

Our approach is compared to the state-of-the-art in Table 1 with the compared results taken from [20] and [13]. Since, as discussed above, the number of embedding dimensions does not have much impact on the other approaches, all results in [20] and [13] are reported using 64 dimensions. For fair comparison, we report our results at 64 dimensions, but also at the better performing higher dimensions. Our approach outperforms the other methods in both the NMI and Recall@K measures, at all embedding sizes presented. Our approach is able to produce better compact embeddings than existing methods, but can also take advantage of a larger embedding space. Figure 4 shows a t-SNE [25] visualisation of the Birds200 test set embedding space. Despite the test classes being withheld during training, bird species are well clustered.

4.2 Image Classification

Experimental Set-up

We evaluate classification performance on the object recognition dataset Caltech-256 [5] and the fine-grained recognition dataset CUB-200-2011 (Birds200) [28]. Unlike the metric learning problem in Section 4.1, the networks are trained and evaluated on the same classes. The standard data splits are used for Birds200, with 5,994 training images. Since we train our own baseline, we crop the Birds200 dataset using the provided bounding boxes, raising both the baseline and our performance (note that we do not crop images in Section 4.1). Caltech-256 consists of a
Table 1: Metric learning results.

| Dims  | Cars196 Dataset | Birds200 Dataset |
|-------|----------------|------------------|
|       | NMI  | R@1 | R@2 | R@4 | R@8 | NMI  | R@1 | R@2 | R@4 | R@8 |
|-------|------|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|
| Semi-hard [17] | 64   | 53.35 | 51.54 | 63.78 | 73.52 | 82.41 | 55.38 | 42.59 | 55.03 | 66.44 | 77.23 |
| LiftStruct [21]  | 64   | 56.88 | 52.98 | 65.70 | 76.01 | 84.27 | 56.50 | 43.57 | 56.55 | 68.59 | 79.63 |
| N-pairs [19]    | 64   | 57.79 | 53.90 | 66.76 | 77.75 | 86.35 | 57.24 | 45.37 | 58.41 | 69.51 | 79.49 |
| Triplet/Gbl [12] | 64   | 58.20 | 61.41 | 72.51 | 81.75 | 88.39 | 58.61 | 49.04 | 60.97 | 72.33 | 81.85 |
| Clustering [20] | 64   | 59.04 | 58.11 | 70.64 | 80.27 | 87.81 | 59.23 | 48.18 | 61.44 | 71.83 | 81.92 |
| SmartMine [13]  | 64   | 59.50 | 64.65 | 76.20 | 84.23 | 90.19 | 59.90 | 49.78 | 62.34 | 74.05 | 83.31 |
| RBF (Ours)      | 64   | 62.15 | 71.05 | 80.74 | 88.06 | 92.79 | 61.26 | 51.15 | 64.64 | 75.57 | 84.72 |
| RBF (Ours)      | 128  | 63.35 | 73.52 | 83.37 | 89.80 | 93.76 | 61.72 | 52.08 | 64.69 | 76.05 | 84.86 |
| RBF (Ours)      | 256  | 63.76 | 77.35 | 85.49 | 91.10 | 94.81 | 62.18 | 54.74 | 67.18 | 77.53 | 86.09 |
| RBF (Ours)      | 512  | 64.68 | 78.39 | 86.91 | 92.06 | 95.52 | 63.50 | 55.91 | 68.26 | 78.63 | 86.38 |
| RBF (Ours)      | 1024 | **65.30** | **79.65** | **87.33** | **92.36** | **95.65** | **63.95** | **57.22** | **68.75** | **79.12** | **87.14** |

Figure 4: Visualisation of the CUB-200-2011 test set embedding space, using the t-SNE algorithm [25]. Despite not being trained on the test classes, bird species are well clustered. Best viewed in colour and zoomed in on a monitor.

minimum of 80 images per 256 object classes. We take 30 images per class as the test set and evaluate at different numbers of training images per class. All images are resized to 256x256.

Our RBF classifier/solver is compared to a conventional softmax classifier/solver. Three base network architectures are considered: AlexNet [11], VGG16 [13] and ResNet50 [8]. The RBF results are obtained by removing the softmax layer and placing an RBF solver after an embedding layer. For AlexNet and VGG, we use FC7 as the embedding layer, creating a 4096 dimension embedding space. Note that we do not include a ReLU layer after FC7. For ResNet, we take the final average pooling layer as the embedding layer, creating a 2048 dimension embedding space. We find that following the ResNet embedding layer with a dropout layer results in a small performance gain. For
fair comparison, we also include this dropout layer in the softmax ResNet configuration, similarly resulting in a small performance gain. We use a weight decay of 0.0005, momentum of 0.9 and train until convergence. A base learning rate of 0.001 is used for softmax and 0.00001 for RBF. We use 500 nearest neighbours from FANNG. Networks are initialised with ImageNet pre-trained weights.

**Image Classification on Caltech-256**  Figure 5 shows the test set classification accuracy of our RBF solver and a conventional softmax solver, both with a base network of VGG16, on the Caltech-256 dataset. The performance is evaluated at varying numbers of training images per class. The test set is the same for each experiment and contains 30 images per class. A random subset of images is used for the training sets, with the same training images used for the RBF and softmax experiments. Our RBF solver outperforms the softmax solver at all numbers of training images, with the performance gain particularly large when training data is scarce.

**Fine-grained Recognition on Birds200**  Image classification accuracy is reported in Table 2 for the Birds200 test set. The results compare conventional softmax solvers to our RBF approach, with various network architectures. Our approach outperforms the softmax counterpart for each network. The performance gain over softmax is larger for AlexNet and VGG than for ResNet. This is likely because ResNet has significantly more non-linear activation function layers than AlexNet and VGG, meaning there is less improvement seen when using the highly non-linear RBF solver.

**5 Conclusion**

Our novel nearest neighbour radial basis function solver is able to be directly applied to both classification and embedding learning tasks. When used as a classifier, our solver outperforms a conventional softmax solver in terms of classification accuracy on the object recognition dataset Caltech-256 and the fine-grained recognition dataset CUB-200-2011. In the domain of embedding learning, or distance metric learning, our approach learns better embeddings than existing deep metric learning approaches, outperforming current state-of-the-art on the Stanford Cars196 and CUB-200-2011 datasets.
Acknowledgments

This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

[1] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a Siamese time delay neural network. In Advances in neural information processing systems (NIPS 1993), 1993.

[2] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

[3] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), volume 1, pages 539–546, 2005.

[4] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In Jmlr, volume 32, pages 647–655, 2014.

[5] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ‘06), volume 2, pages 1735–1742, 2006.

[7] B. Harwood and T. Drummond. FANNG: Fast Approximate Nearest Neighbour Graphs. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5713–5722, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[9] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In International Workshop on Similarity-Based Pattern Recognition, pages 84–92, 2015.

[10] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 554–561, 2013.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, pages 1097–1105. 2012.

[12] V. B. G. Kumar, G. Carneiro, and I. Reid. Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimizing Global Loss Functions. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5385–5394, 2016.

[13] V. B. G. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond. Smart Mining for Deep Metric Learning. arXiv preprint arXiv:1704.01285, 2017.

[14] C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval, volume 1. Cambridge university press Cambridge, 2008.

[15] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 512–519, 2014.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015.
[17] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 815–823, 2015.

[18] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556, 2014.

[19] K. Sohn. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In Advances in Neural Information Processing Systems 29, pages 1857–1865. 2016.

[20] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Learnable Structured Clustering Framework for Deep Metric Learning. arXiv preprint arXiv:1612.01213, 2016.

[21] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4004–4012, 2016.

[22] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015.

[24] Y. Tang. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.

[25] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

[26] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning Fine-Grained Image Similarity with Deep Ranking. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 1386–1393, 2014.

[27] K. Q. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. Advances in neural information processing systems, 2006.

[28] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.