ResFuse and ReFiNet: Enhanced CNN Architectures for Image Classification

Nihit Desai *
Stanford University
nihit@stanford.edu

Frederic Ren *
Stanford University
fren@stanford.edu

Allen Nie *
Stanford University
anie@stanford.edu

Abstract

Convolutional Neural Networks are the key components for many state-of-the-art models in various computer vision tasks Recent work in this area suggests that network depth is an important factor in achieving good results. In this paper we propose two improvements to deep CNN architectures and evaluate performance on image classification: ResFuse (a modified Deep Residual Network with a projected convolution connecting input and output between two residual blocks) and RefiNet (a CNN-based hierarchical classification framework). In experiments, we show that ResFuse performs 1-3% better than reference network, and ReifiNet performs 2-3% better than the baseline.

1. Introduction

Visual recognition is a well studied problem in Computer Vision. Feature descriptors such as HOG [3] and SIFT [20] have been widely used to extract features from images. However, most hand-engineered features suffer from the following drawbacks - they are too sensitive to input, don't generalize well and are limited in their representational power. Deep Convolutional Neural Network (CNN) architectures[17] have recently become popular as they eliminate the dependence on hand-designed explicit features and instead directly learn good feature representations from raw data. These deep neural networks integrate features at various levels of specificity (ranging from low-level features like edges to high-level abstract features such as objects)[30] and classifiers in a comprehensive end-to-end fashion. Indeed, Deep CNN architectures have achieved state-of-the-art results on image classification tasks[16], [12].

Recent work in this area suggests that network depth is very important to get good results since deeper networks can learn more abstract features[22]. Deep Residual Networks[11] (which advanced the state of the art performance on image classification and achieved a top-5 error rate of 3.57% in ILSVRC 2015) allow much deeper networks to be trained efficiently and without running into the problem of vanishing/exploding gradients which plagues vanilla CNNs as depth of the network increases. Another interesting line of work on improving CNN architectures has been to perform image classification in a hierarchical fashion using CNNs. As the number of object categories in a dataset becomes larger, "separability" between different object categories is highly uneven. Some pairs of categories are much harder to distinguish than others, e.g. in ImageNet dataset, the categories 'iPod' and 'German shephard' are easier to distinguish that 'sulpher butterfly' and 'monarch butterfly'. Hierarchical classification has been effective for linear classifiers [6] [31], and some recent work has tried to formulate such an approach to CNNs [28] [4].

In this project we explore two possible improvements to CNN architectures. We propose ResFuse network, which explores the efficacy of adding redundant connections between residual blocks [11]. We also propose ReifiNet, which extends the Deep Convolutional Neural Network architecture to perform image classification in a hierarchical fashion. To evaluate our improvements, we use the two datasets (both of which are subsets of the ImageNet dataset): ImageNet-100 (50K images across 100 classes) and ImageNet-200 (100K images across 200 classes). In the context of this class, these datasets have also been referred to as Tiny-ImageNet-A and Tiny-ImageNet. We compare the performance of our architectures on image classification task against relevant baselines. A more detailed explanation of our improvements follows in Section 4 and 5.

2. Related Work

Convolutional Neural Networks: CNN-based models produce state-of-the-art performance in various computer vision tasks, including image classification [16][11], object detection [7][10], and semantic segmentation [2]. There has been considerable work in enhancing CNN components, including pooling layers [29], activation units [8], and

*equal contribution

nonlinear layers[18]. There has also been considerable work in improving CNN training, such as by using Batch Normalization[14] and Dropout[23].

Shortcut Connection Networks: Different enhanced CNN architectures that focus on shortcut connections have been proposed over the recent years, two of which are Highway Network, Inception and Residual Network. Highway Network is developed by Srivastava [25]. It took inspiration from Long-short-term Memory Network, using two convolutions as gating units. It claims the highway connections help training very deep networks, and helping with the vanishing gradient problems [26]. In [27], an "Inception" block was proposed and the core concept of an inception block is to allow input to pass through different branches and combine computations in the end. Another CNN architecture has been gaining a lot of attention is Residual Network. Residual Network, similar to its predecessor Highway Network, and Inception, is a network that focuses on shortcut connections. It has demonstrated to have exceptional performance in deep networks.

Hierarchical CNNs: In visual recognition, there is a vast literature exploiting category hierarchical structures. Hierarchical classification has been shown to be effective for linear classifiers [6] [31] in this problem domain. There are two ways that existing approaches construct the label hierarchy - either predefined manually (e.g. [5] uses the predefined ImageNet hierarchy) or learned automatically from data[19]. An early attempt to introduce a category hierarchy in CNN models is reported in [4], where label relations are encoded in a hierarchy and improved accuracy is achieved when a subset of training images are relabeled with internal nodes in the hierarchical class tree. More recently Yan *et. al.*'s work[28] proposes a hierarchical approach to image classification using CNNs. They construct a two-level category hierarchy from ImageNet-1000 dataset and report an improvement in classification accuracy compared to a single layer flat CNN of comparable size. Our work in this regard closely follows Yan *et. al.*'s work.

3. Dataset

To evaluate the performance of our new architectures on image classification tasks, we consider two datasets: ImageNet-100 (50K images across 100 classes) and ImageNet-200 (100K images across 200 classes). In the context of this class, these datasets have also been referred to as Tiny-ImageNet-A and Tiny-ImageNet. We use ImageNet-100 to evaluate both the architectures, and additionally use ImageNet-200 to evaluate RefiNets. It should be noted that the focus of our project is to evaluate the proposed architectures against corresponding baselines on these datasets, **not participating in the ImageNet challenge**. Each image in this dataset has a spatial dimension of 64x64x3. Our RefiNet architecture resizes and preprocesses these images (finally converted to spatial dimension 224x224x3). A brief summary of the datasets is given in

	ImageNet-200	ImageNet-100
Training	100000	50000
Test	5000	2500
Validation	5000	2500
Classes	200	100

Table 1: Dataset Size

Table 1, and images from some example classes are given in Figure 1. It should be noted that the given datasets have class labels only for the training and validation data, not for the test data. Hence, we divide the validation set into two parts: one used as the actual validation set (to measure performance during hyperparameter tuning and model selection) and another used as the test set (which we don't touch until the very end to report final performance numbers). This is why the size of validation and test sets in Table 1 is lesser than the size of the respective sets in the released dataset.

Figure 1: Sample images from ImageNet-100

4. ResFuse

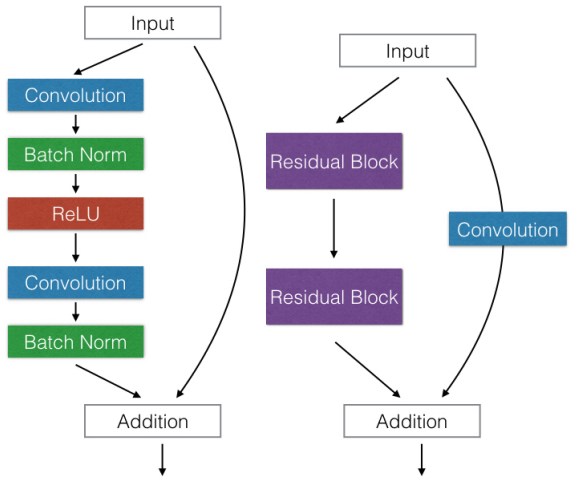
4.1. Architectures

4.1.1 Residual Network

Residual network introduced by [11] employs residual blocks that allow additive interaction between input and output of two convolution layers. The dimension of the input and dimension of the output within the residual unit need not be the same. The gradient is distributed according to the additive operation, and so it partly solves the vanishing gradient problem. The residual unit is described as in equation 1.

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{1}$$

Residual block also has variations when it comes to dimension increase, with regard to an increased number of filters. Residual network alternates on two variations of residual units: zero-padding or weighted projection. In our experimental setting, we used vanilla padded projection.



(a) ResNet Architecture (b) ResFuse Architecture

Figure 2

4.1.2 Highway Network

Highway network is inspired by long-short-term memory gating units of the recurrent neural network. The input passes through two gates, one is the transformation gate, and another is the carry gate, and it also passes through a normal convolution layer. In the end, the multiplicative operation combines the result from the vanilla convolution, transformation gate, and the carry gate. Transformation gate defines how much "transformation" through the vanilla convolution it allows to pass through, and carry gate decides how much of the original input the network decides to carry. All of which are expressed in Equation 2.

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C) \quad (2)$$

However, in order to simplify computation and reduce the number of needed parameters, the paper chooses to follow the convention that reduces the carry gate to an operation of $1 - T(x, W_T)$, so the final equation is Equation 3.

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T)) \quad (3)$$

Then we can make the argument if the network decides not to take any of the original input, transformation gate would learn to be an identity projection, thus blocking any original input, or if it believes the convolution is not helpful, it can skip the transformation and directly send input to the next layer. This intuition is captured in Equation 4. Residual network can be considered as a special version of the highway network, with both transformation and carry gate as the identity projection.

$$y = \begin{cases} x, & \text{if } T(x, W_T) = 0, \\ H(x, W_H) & \text{if } T(x, W_T) = 1. \end{cases} \quad (4)$$

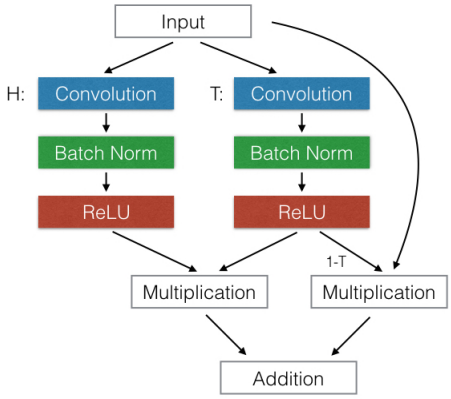


Figure 3: Highway Network Architecture

4.1.3 ResFuse Network

We propose a new architecture: ResFuse network. It takes inspiration from probabilistic graphical modeling that any probabilistic network is under conditional independence assumption, and probabilistic models simplify the connections based on conditional independence assumption.
Even though it is a conditional independence assumption under the generative model, and neural networks are in general considered a discriminative model, similar phenomenons can observed in dropout network [24], and model ensemble methods [9].
We propose a neural network with redundant projections that are built on top of residual network's simple additive connections. We connect input and convoluted output on the scope of two residual blocks (4 convolution layers), and introduce such redundant connections throughout the entire network.

$$y = \mathcal{F}(\mathcal{F}(x) + x) + C(x) \quad (5)$$

We define a carry gate: C, a convolution transformation that is composed of 1x1 filter size, with stride of 1. We allow the weights of C to be learned and if such redundant connection is not needed, it can equate to the zero-matrix. Similar to highway network, we do not formulate a version where input and output can be of different size, and such dimension increase is handled by a regular residual block between the resfuse blocks.

4.2. Experiments & Results

We evaluate our method on the ImageNet-100. We evaluate variations of residual network, highway network, and resfuse network with different number of layers and obtain a final test result on a test set. We evaluate both top-1 and top-5 error rates during training, and top-1 error rate on testing. Due to the time constraint, we adopted part of our hyperparameters from [11], and conducted a limited hyperparameter search mostly on the number of training epochs, number

of layers, and learning rate drop. It turns out these three factors governed the performance of our models.

We constructed a classical ResNet-34 layer model as our reference model, then we built all three networks with 19 layers, and we built ResNet and ResFuse net to 25 layers. All models with the same depth (except for the classical model) share the same number of convolutions, and the same filter size, stride, and padding strategy.

We chose nesterov accelerated gradient (NAG) [21] as our optimizing strategy, as stated in [11], other optimization methods such as Adam, perform poorly in the context of ResNet. This seems to be a common trait of the shortcut connected networks, which is also mentioned in highway network. All the weights are initialized using He normalization [13]. We initialize the weight and bias of highway gating units the same as described in the paper [25].

Then we ran the experiments on GPU, and we configured the batch size to be the maximum number of images that can be stored in the GPU memory. The training loss is plotted in Figure 4 for all models. We drop our learning rate by 0.1 when we encounter a training plateau, and decision is made based on observation. All models during training phase converge to the same loss, but highway network performed the best in validation.

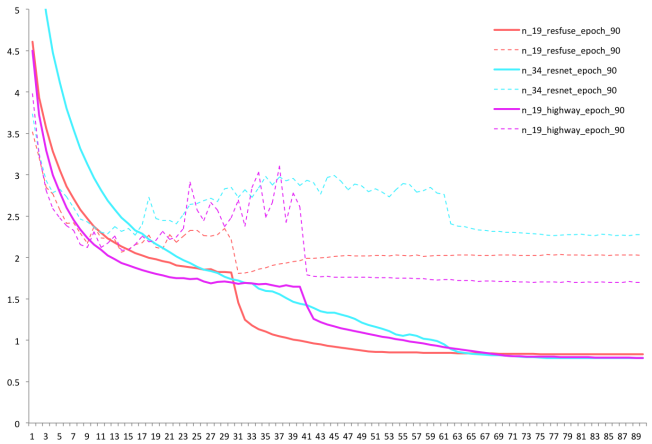


Figure 4: Training loss aof architectures

We also provide a comparison of model's top-1 and top-5 result in Table 2. All experiments are run at 90 epochs, and the testing result is obtained on the entire validation set.

4.3. Discussion

We can notice from Table 2 that clearly 34-layer ResNet model is too deep for the size of the ImageNet-100 dataset, and it is underperforming in validation set compared to other models. Judged from comparison, ResFuse consistently bests the performance of ResNet by a small margin with the same depth and training configuration. However,

Model	Val top-1	Val top-5	Test top-1
ResNet*	57.70	79.86	57.55
ResNet-19	60.20	82.84	60.18
Highway-19	62.50	**84.30**	62.48
ResFuse-19	60.48	**83.24**	60.36
ResNet-25	60.06	82.42	60.06
ResFuse-25	60.32	82.74	60.16

Table 2: Models training accuracy (in %) on ImageNet-100. *: reference resnet is a 34-layer classical model

evidence does support that Highway network performs better than both ResFuse and ResNet.

We have also compared the training time and size increase between the three architectures, which is shown in Table 3. ResFuse is taking a time penalty due to the added redundant connection with a slight parameter increase. The amount of parameter increase is subject to the number of filters of the previous layer, since ResFuse projection is only doing a simply identity projection.

ResFuse also shares the problem of highway network: both networks cannot handle dimension increase between the input and the final projected output. Highway network uses normal convolution layers between highway blocks to increase the number of filters, and ResFuse Network uses a residual block to increase the filter size.

	Parameter Size	Time/Epoch
ResNet-19	423,5172	6.470m
Highway-19	423,5172	6.617m
ResFuse-19	432,2084	6.803m

Table 3: Models training accuracy (in %) on ImageNet-100. *: reference resnet is a 34-layer classical model

5. RefiNets

5.1. Architecture

In this section, we discuss the details of various parts of the RefiNet architecture. An overview of this architecture is given in Figure 5. Overall goal of RefiNet is to perform image classification in a hierarchical manner. In the rest of the section, we will use the following notations. A dataset consists of images $\{x_i, y_i\}$, where x_i and y_i denote the image data and label, respectively. Size of the dataset is N, there are C class labels which we intend to cluster into K clusters.As a preprocessing step, we subtract channel mean, resize input images with spatial dimensions 64x64 to 256x256 and maintain a central crop of 224x224. We then reorder color channels to BGR. Some of these steps are required because we intend to use a pretrained model (transfer learning) in Level 1 of our architecture. Our

RefiNet work closely follows the work of Yan *et. al.*[28] with changes mainly to how we learn the label hierarchy, and architecture of level-2 CNNs.

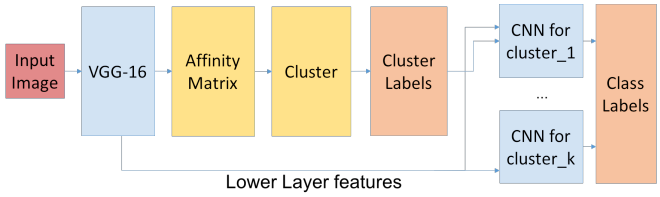


Figure 5: RefiNet architecture

VGG-16: VGG Networks (first proposed by [22]) have become fairly popular due to their simple and easy to understand architecture and good performance. For the first level of classification of our architecture, we use a 16 layer VGGNet pretrained on ImageNet-1000 dataset. Since our dataset has 100 or 200 class labels (depending on whether ImageNet-100 or ImageNet-200), we retrain only the last two layers for this level and freeze all the previous layers, as given in Figure 6.

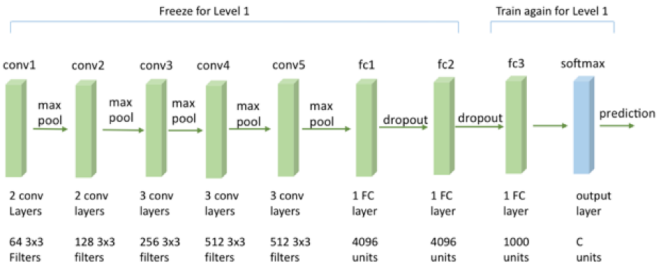


Figure 6: VGG-16 transfer learning
Source: figure adapted from [28]

Constructing class label hierarchy: Our goal of building a class label hierarchy is to group together class labels that are often confused with each other into the same cluster. In a later stage, a dedicated CNN will be trained to distinguish images among these confused classes. After training of VGG-16 network in Level 1 is completed, we sample a subset of the validation set (of N_1 images) and get class prediction scores from VGG-16. Using these scores, we define a pairwise affinity score for class labels and cluster the labels based on this score ($A_{p,q}$ is the affinity between pair of classes p and q):

$$A_{p,q} = \frac{1}{N_1}\left(\sum_{x_i:y_i=p} score_{x_i,q} + \sum_{x_i:y_i=q} score_{x_i,p}\right) \quad (6)$$

Intuitively, Equation 6 tries to capture "For a pair of labels (p, q), how much probability mass is shifted from p to q and vice-versa". An alternative we also tried was to use the confusion matrix directly: given the normalized CxC confusion matrix **F**, we defined affinity matrix as:

$$A = \frac{1}{2}\left(F + F^T\right) \quad (7)$$

However, we found that first approach works better, most likely since the confusion matrix is very sparse. We think that given a large enough validation set from which a confusion matrix is constructed, both approaches are likely to work equally well. We tried a few different clustering algorithms and observed that spectral clustering (followed by a K-Means in the low dimensional space) performs best for our usecase. Performance of various clustering algorithms is given in Table 4.

This procedure, which clusters a given set of class labels, can be repeated in a recursive manner to construct a deeper hierarchy. We can conceptually think of the results of this clustering as a mapping $P : [1, C] \rightarrow [1, K]$. We append this class label to cluster mapping aggregation layer which maps the class predictions of Level-1 VGG-16 network into cluster labels for each input image. This aggregation mapping layer uses a voting mechanism: the top 5 predicted class labels each maps to a corresponding cluster, we count the number of votes each cluster has among the top 5 predictions and assign the cluster label with most votes as the predicted cluster label for the given image. These cluster labels will enable conditional executions of CNNs at the next level.

Level-2 CNNs: At level-2 we train a set of independent CNN models, one for each cluster $\{M_i\}_{i=1}^{i=K}$. Each such CNN makes class label predictions and is trained only on subset of the training data that belongs to class labels in the given cluster. Hence each level-2 CNN only excels in classifying a small set of class labels and produces a prediction over a partial set of categories. The probabilities of other class labels absent in the partial set are implicitly set to zero.

Each level-2 CNN shares parameters of the lower layers of level-1 pretrained VGG-16 network. Alternatively, we can say that lower layers of VGG-16 are used to extract features from raw images for level-2 CNNs. More Specifically, we reuse the first 10 layers of pre-trained VGG-16 (i.e. up to the output of "conv4" in Figure 6) and **do not** backpropagate into these layers when training the level-2 CNN. The reason for this is that preceding layers in deep networks response to class-agnostic low-level features such as corners and edges. Since such features are useful irrespective of the class labels, sharing these lower layer parameters avoids extracting these features twice over. Secondly, it reduces the total number of parameters and speeds up level-2 training. Each level-2 CNN then

trains the later layers from scratch. Since the later layers are trainined individually for each level-2 CNN, it allows these CNNs to extract more subtle and nuanced features to distinguish between class labels that level-1 CNN is more likely to confuse. If desirable, we can repeat this procedure to construct a deeper hierarchy by repeating the clustering process at one of the level-2 CNNs and then have level-3 CNNs for each of the clusters, and so on. Architecture of level-2 CNNs is outlined in Figure 7.

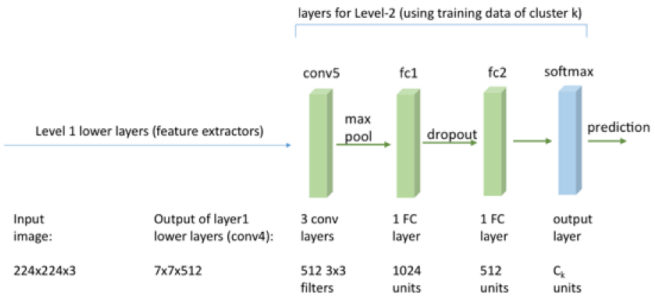


Figure 7: CNN architecture on Level-2

5.2. Experiments & Results

We evaluate RefiNet architecture on ImageNet-100 and ImageNet-200, and evaluate its performance compared to the single-level VGG-16 network. We also perform some qualitative evaluations on clusters and class-specific performance (which classes perform significantly better or worse in the new approach compared to VGG-16).

5.2.1 Training overview

Our training boradly contains the following steps:

- Train the level-1 VGG network, hyperparameter tuning, evaluation.
- Construct CxC affinity matrix, and cluster labels to learn label hierarchy
- Train level-2 CNNs, evaluation

To train our networks, we use Adam optimization algorithm[15]. For hyperparameters related to Adam, the default values of beta1 and beta2 worked well, whereas we performed a random search for a good **learning rate** in the range $[10^{-2}, 10^{-6}]$. Learning rates are chosen individually for level-1 and level-2 CNNs. For hyperparameters related to network architecture, we use the following values: **filter size: 3x3**, **dropout: 0.5**, **maxpool: 2x2**. These architectural parameters are maintained across level-1 and level-2 CNNs. For all steps in the training process, we used g2.2xlarge GPU instances (preconfigred AMIs shared by the class instructors) available on Amazon AWS. These instances have 15GB of memory and 4GB of GPU memory. RefiNets are implemented using Theano[1] and Lasagne. Our mini-batch size was mainly a function of the amount of GPU memory and number of model parameters. We used a **mini-batch size of 125**.

5.2.2 Label hierarchy

We experimented with various clustering techniques, including agglomerative clustering, where each class starts as a unique cluster and during each iteration the pair of clusters closest to each other are merged; affinity propagation, where classes are selected based on mutual similarity to be the "exemplar" to represent the cluster they belong to; K-Medoids, a variation of K-Means clustering where medians instead of means are used in selecting new centers; and spectral clustering explained in Equation 6. Since the cluster labels will enable conditional executions of CNNs at the next level, the most important metric to evaluate the quality of this clustering in our case is the cluster label accuracy (% of validation examples for which the cluster of the true label matches the cluster of the predicted label). Results for various clustering techniques are shown in Table 4, we see that spectral clustering give the best accuracy of 94.1%.

Clustering algorithm	Cluster accuracy
Agglomerative	79.8%
Affinity Propagation	81.0%
K-medoids	85.0%
Spectral clustering	**94.1%**

Table 4: ImageNet-100: Cluster label accuracy on Level-1

We also explored the relationship between cluster accuracy and the number of clusters, starting with $k = 4$. As shown in Figure 9a, accuracy is highest when $k = 5$, and decreases as the number of cluster increases. Figure 9b shows the division of classes into 5 clusters, and Figure 8 is a qualitative evaluation of class labels within each cluster for ImageNet-100. One can see that the learned hierarchy is intuitively easy to understand - classes within each cluster are indeed quite related, e.g. cluster#1 contains all insects and similar creatures (butterflies, cockroach, tarantula) , cluster#2 contains natural structures (cliff, lakeside, coast) and so on. Notice that $C1$ has 41 member classes, we hypothesized that this relatively big cluster could have had an adverse impact on overall accuracy, and we repeated the cluster-retrain procedure on this cluster, splitting it further into two more clusters, as shown in Figure 8. The overall classification accuracy of RefiNet is discussed in Section 5.3.

5.2.3 RefiNets

Once we have constructed the label hierarchy, we evaluate RefiNet performance, as part of which we compare the following approaches:

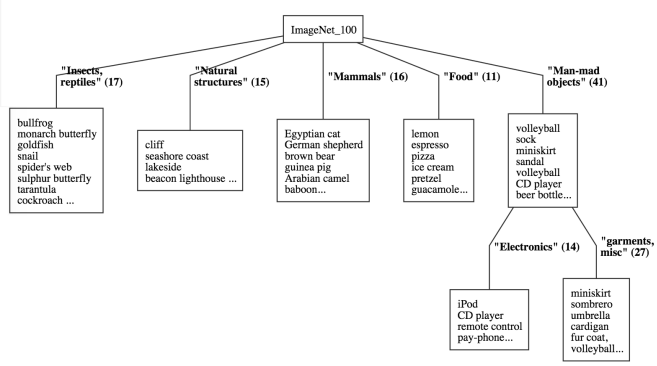
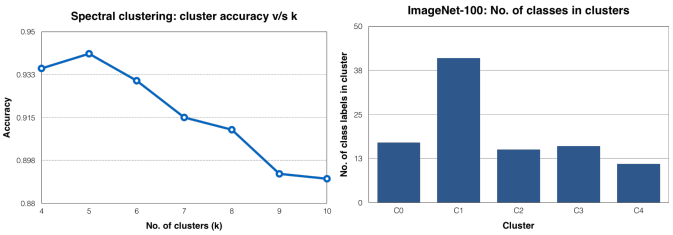


Figure 8: Learned clusters are intuitively easy to interpret



(a) Cluster accuracy v/s k (b) Cluster sizes for k=5

Figure 9: Analysis of spectral clustering on ImageNet-100

- **VGG-16**: This is our baseline model, pretrained VGG-16 on ImageNet, with retrained last two layers.
- **RefiNet-L2**: This is our RefiNet model with two levels of hierarchy - we use the pretrained VGG-16 network on level-1, use spectral clustering to contruct label hierarchy, and use level-2 CNNs which share parameters with level-1 as outlined in the previous section.
- **RefiNet-L2 with retraining level-1 to output cluster labels directly**: This is similar to RefiNet-L2, but instead of using the label hierarchy in a post-forward pass step to predict cluster labels from class labels in Level-1, we retrain the output layer of Level-1 CNN to directly classify an image into one of the cluster labels.
- **RefiNet-L3**: This is similar to Refinet-L2, but we go one level deeper. More specifically, we recursively cluster level-2 labels belonging to the largest cluster and train level-3 CNNs

To measure the performance for the entire dataset, we use two quantitative metrics (to measure class-specific performance, we will use a few other metrics discussed later): **top-1 accuracy** (% of examples for which the true class label matches the top predicted label) and **top-5 accuracy** (% of examples for which the true class label is in the top-5 predicted label). Training proceeds using Adam optimization as discussed in Section 5.2, and an analysis of training loss as a function of number of epochs is given in Figure 10.

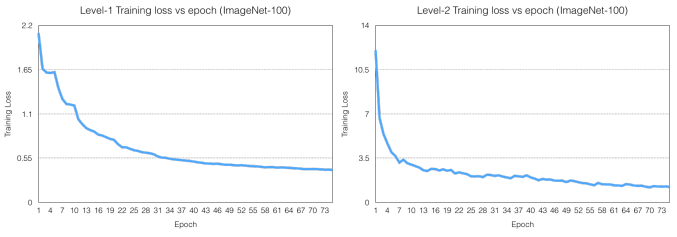
Performance of these approaches on ImageNet-100 is summarized in Table 5 and ImageNet-200 is summarized in Table 6. Compared to VGG-16, our best performing model achieves an improvement of close to **2%** in top-1 accuracy for ImageNet-100 and **2.5%** in top-1 accuracy for ImageNet-200 on the validation set. Gains in top-5 accuracy are more modest, and in Section 5.3 we discuss some possible reasons. As noted in Section 3, we split the original validation set into 50% validation set (used for hyperparameter tuning) and 50% test set. Gains on the test set are quite comparable which might indicate the gains are generalizable.

Model	**Val top-1**	**Val top-5**	**Test top-1**	**Test top-5**
VGG-16	70.60	90.04	70.81	91.22
RefiNet-L2	**72.50**	**90.24**	**72.18**	**90.34**
RefiNet-L2 + retrain	70.41	88.72	69.59	90.81
RefiNet-L3	**72.78**	**90.21**	**72.89**	**91.30**

Table 5: Model accuracies (in %) on ImageNet-100

Model	**Val top-1**	**Val top-5**	**Test top-1**	**Test top-5**
VGG-16	60.81	81.24	60.93	80.79
RefiNet-L2	**62.92**	**82.08**	**62.40**	**81.96**
RefiNet-L2 + retrain	61.74	80.71	60.35	80.25
RefiNet-L3	**63.27**	**82.67**	**63.38**	**82.03**

Table 6: Model accuracies (in %) on ImageNet-200



(a) Level-1 VGG-16 CNN (b) Level-2 CNN for one of the clusters

Figure 10: Training loss as a function of epochs

5.3. Discussion

In this section, we discuss and analyze results of our experiments. As shown in the last section, RefiNet-L3 (our best performing model) attained roughly 2% improvement in top-1 accuracy for ImageNet-100 and 2.5% for ImageNet-200. Moreover, when we further split the largest cluster obtained in RefiNet-L2 into two new clusters and repeated the procedure (RefiNet-L3),the top-5 accuracy

on ImageNet-100 did not improve at all, whereas for ImageNet-200 it improved by 0.4%. This might indicate that RefiNet (and hierarchical classification in general) performance improvement compared to baseline will be larger for datasets that are sufficiently large and contain greater number of classes. As the number of classes increase, it becomes increasingly difficult for a flat classifier to model it and learn all the subtle features that help it distinguish closely related pairs of classes. In such a scenario, hierarchical classification can help.

It is interesting to observe the discrepancy of the changes in top-1 and top-5 accuracies. On the one hand, the initial assumption is that the classes within the same cluster should be preferred over those outside of the cluster, as they are more similar to the true label; on the other hand, since most of the classes not in the current cluster is excluded from the training data during L2 training, only being able to choose from within the cluster might have been too restrictive. Table 7 shows the performance of level-2 CNNs on ImageNet-100. This performance is measured only by considering the subset of the data that belongs to class labels in the given cluster. We see an improvement across the board, when comparing performance to level-1 pretrained VGG-16. However, what limits the performance improvement of the entire architecture is the cluster accuracy - if cluster label assignment for an image is incorrect on level-1, there is no way to "recover" from this mistake in subsequent level. We discuss a way to potentially extend RefiNets to overcome this limitation in Section 6. One additional insight we would like to share is the relationship between clusters of class labels learned by our approach and t-SNE embedding visualization of our dataset. We replaced the class labels of images in the validation set with the cluster labes and visualized this dataset (with color of the datapoint indicating cluster label) using t-SNE embedding, results of which are shown in Figure 11. We see the clusters separated fairly well in this embedding which further gives us confidence that the clustering is meaningful.

Cluster	size	Val top-1	Val top-5
1	17	81.13	98.62
2	41	70.36	91.55
3	15	85.39	97.61
4	16	79.83	98.02
5	11	82.00	99.27

Table 7: level-2 CNNs accuracy (in %) on ImageNet-100

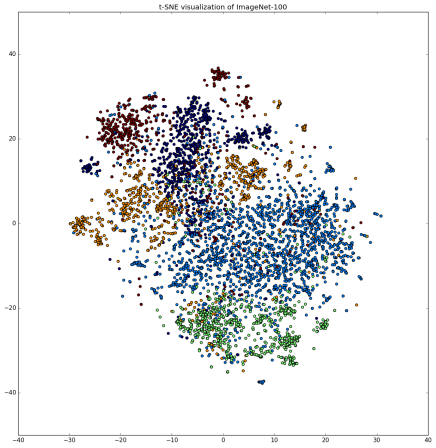


Figure 11: t-SNE Embedding of ImageNet-100

6. Future Work

6.1. ResFuse Network

We only explored a redundant connection for every two residual blocks. We can introduce an additional redundant connection for three blocks as well, and until we get a fully connected redundant network. Exploring the trade-off between time per epoch and accuracy increase through redundant connection increase could be another possible focus of study. We also would like to compare ResFuse with ResNet that utilizes convolution-projection based dimension increase, to see if two networks perform similarly or if ResFuse still maintains similar margins over ResNet.

6.2. RefiNet

Our RefiNet-L3 model was the best performing and attained roughly 2% improvement in top-1 accuracy for ImageNet-100 and 2.5% for ImageNet-200. The final classification accuracy of RefiNet is highly dependent on the clustering accuracy, which is significantly lower than 100%. In stead of assigning each class to strictly one cluster, we use a softer clustering technique, allowing it to pick from multiple clusters, weighting the final class scores by initial predictions.

Since RefiNet typically works better with datasets that has more classes, we propose to recursively run RefiNet on the full ImageNet dataset with 1000 classes. At each iteration the algorithm would split the largest cluster into two clusters and perform classification on these two new clusters. The algorithm stops when all individual cluster has size smaller than some threshold.

References

[1] J. Bergstra, O. Breuleux, and et. al. Theano: A cpu and gpu math expression compiler. *Python for Scientific Computing Conference*, 2010.

[2] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *arXiv technical report arXiv:1512.04412*, 2015.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *in CVPR*, pages 886–893, 2005.

[4] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. *ECCV*, 2014.

[5] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. *CVPR*, 2012.

[6] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic label sharing for learning with many categories. *ECCV*, 2010.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[8] I. Goodfellow, D. Warde-Farley, M. Mirza, and A. Courville. Maxout networks. *ICML*, 2013.

[9] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001, 1990.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *ECCV*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *in ArXiv technical report arXiv:1512.03385*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *JLMR*, 2015.

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *in NIPS*, 2012.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[18] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, 2013.

[19] B. Liu, F. Sadeghi, M. Tappen, O. Shamir, and C. Liu. Probabilistic label trees for efficient large scale image classification. *CVPR*, 2013.

[20] D. Lowe. Distinctive image features from scale-invariant keypoints. *in IJCV*, 2004.

[21] Y. Nesterov et al. Gradient methods for minimizing composite objective function. Technical report, UCL, 2007.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JLMR*, 2014.

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

[25] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[26] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, 2015.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *ICPR*, 2015.

[28] Z. Yan, H. Zhang, R. Piramuthu, and V. Jagadeesh. Hd-cnn: Hierarchical deep convolutional neural network for large scale visual recognition. *ICCV*, 2015.

[29] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *ICLR*, 2013.

[30] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. *ECCV*, 2014.

[31] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. *ICCV*, 2007.