Pre-trained CNNs as Visual Feature Extractors: A Broad Evaluation

Andrew Holliday[1] and Gregory Dudek[2]

***Abstract*—In this work, we perform a wide-ranging evaluation of Convolutional Neural Networks (CNNs) as feature extractors for matching visual features under large changes in appearance, perspective, and visual scale. Our evaluation covers 82 different layers from twelve different CNN architectures belonging to four families: AlexNets, VGG Nets, ResNets, and DenseNets. To our knowledge, this is the most comprehensive analysis of its kind in the literature. We find that the intermediate layers of DenseNets serve as the best feature extractors overall, providing the best overall trade-off of robustness to feature size. Moreover, we find that for each network, the later-intermediate layers provide the best performance, regardless of the total number of layers in the network.**

I. INTRODUCTION

Feature extraction and matching are key parts of many pipelines for vision-based robotics. They are crucial to state-of-the-art techniques for visual odometry and place-recognition, among other things, and are required to be robust to irrelevant changes in perspective and appearance. At present, most such methods rely on human-engineered feature extraction algorithms, like the Scale-Invariant Feature Transform (SIFT) proposed by Lowe et al. [1], or the GIST descriptor proposed by Olivia et al. [2]. However, since the victory of Krizhevsky et al. [3] in the ImageNet classification competition [4] using a Convolutional Neural Network (CNN) architecture, CNNs have come to form the backbone of image processing pipelines for a wide variety of applications, from image classification to object detection, tracking, and place recognition.

In this work, we explore how well pre-trained CNNs perform when re-purposed as feature extractors for visual matching problems. To do this, we leverage Object Landmarks [5], our algorithmic framework for feature extraction and matching with robust performance across significant scale changes. We examine features extracted from 82 layers from 12 different ImageNet-trained CNNs with a variety of architectures, investigating their robustness to variations in perspective, visual scale, and appearance. We consider which layers provide the best performance when incorporated into Object Landmarks, and what trade-offs might exist between them. We also consider whether there is any broad relationship between the depth of a layer in a network and its performance, regardless of network architecture. And we investigate whether the choice of network architecture is itself significant.

[1]Andrew Holliday and Gregory Dudek are with the Center for Intelligent Machines, McGill University, 845 Sherbrooke St, Montreal, Canada ahollid@cim.mcgill.ca, greg.dudek@samsung.com

II. RELATED WORK

A. Evolution of CNN Architectures

While CNNs have been used as far back as 1989, as in LeCun et al. [6], the aforementioned ImageNet-winning AlexNet CNN of Krizhevsky et al. [3] revived interest in CNNs for image processing tasks. Since then, many new CNN architectures have been proposed that build on and improve upon the results of [3]. ImageNet continues to serve as a prominent benchmark against which these new architectures are measured. Simonyan et al. exceeded the performance of [3] with their VGG Net architecture. They were in turn superceded by the Inception architecture of Szegedy et al. [7] and by the ResNets of He et al. [8]. Further improvements on the ImageNet benchmark were reported by Huang et al. with the DenseNet architecture [9], and by Xie et al. with the ResNeXt architecture [10].

More recent work has attempted to design CNN architectures automatically via a learning process. This is known asNeural Architecture Search (NAS). The reinforcement learning-driven process of Zoph et al. [11] produced a network architecture that exceeded the performance of any human-designed CNN on ImageNet. Pursuing a different aim, Tan et al. [12] use a NAS process to produce CNN architectures optimized for the strict resource constraints of mobile devices.

While a full survey of image-classification CNNs architectures is beyond the scope of this paper, those described here encompass the most commonly-used architectures in practice.

B. Pre-trained CNNs as Feature Extractors

As each layer of a neural network learns a function of the outputs of previous layers, deeper layers represent more abstract functions of the input. A CNN trained for semantic image-processing tasks must have outputs that are invariant to semantically-irrelevant changes in lighting, perspective, and context. It is thought that the intermediate layers of such a network may be computing functions that encode some high-level semantic features of the input, which ought to be robust to such variations. This idea motivated work by Sünderhauf et al. [13], [14], as well as Babenko et al. [15], Razavian et al. [16] and Chen et al. [17], all of which investigate the use of intermediate activations of a CNN as feature vectors for image retrieval tasks. [13], [15], and [17] all experiment with features from multiple output layers of the network, but all consider only features from an ImageNet-trained AlexNet or the closely-related OverFeat network [18]. These architectures are far simpler than those which are now widely used for image-processing applications, such as DenseNets and ResNets. In this work, we compare features extracted

from different layers of twelve different CNN architectures on visual odometry and point-matching tasks, allowing us to make comparisons between different networks as well as between different layers of one network.

Other work, such as that of Simo-Serra et al. [19], Kwang Moo et al. [20], DeTone et al. [21], and Ono et al. [22] train CNNs specifically for the task of computing visual feature descriptors for matching. While useful in a point-matching context, these systems are trained to discriminate between very small image regions. We are interested here in the use of CNN features as descriptors of arbitrarily-sized image patches, such as may correspond to objects..This task has more in common with ImageNet-style classification than keypoint matching, so ImageNet pre-training is more appropriate. Furthermore, comparing CNNs trained with different objectives and different data would introduce these as potential confounding factors in our results. For these reasons, we consider only ImageNet-trained CNNs in this work.

III. METHODOLOGY

We evaluate features extracted from CNNs in the context of point-feature matching using Object Landmark features. As described in our previous work [5], Object Landmark features provide superior robustness to changes in perspective and scale in localization than using either CNN features or state-of-the-art point features such as SIFT on their own. In this section we describe Object Landmarks, including the modifications made from the earlier version used in [5].

A. Object Landmarks

A schematic illustration of the object landmark extraction pipeline is provided in Figure 1. First, object proposals are computed from an image. The rectangular image patch corresponding to each proposal's bounding box is extracted and deformed to a square of fixed size, which normalizes for some of the appearance change that will occur under changes in scale. Each square patch is passed through a CNN, and an intermediate activation of the network is taken as the descriptor vector of the object.

Point features such as SIFT or Speeded-up Robust Features (SURF) are also extracted from the original image. The point features contained within the bounding box of each object proposal are designated as the sub-features of that object landmark. As the bounding boxes of object proposals can overlap one another, we allow one point feature to be associated with multiple object landmarks.

An Object Landmark feature thus consists of three components:

- A bounding box in 2D image space, defining the object's location in the image,
- A descriptor vector for the object extracted with a CNN,
- A set of point features on the object, consisting of pixel locations and descriptor vectors.

In our previous work [5], we used the Selective Search algorithm of Uijlings et al. [23] to propose bounding boxes, but in this work, we instead use the Edge Boxes algorithm

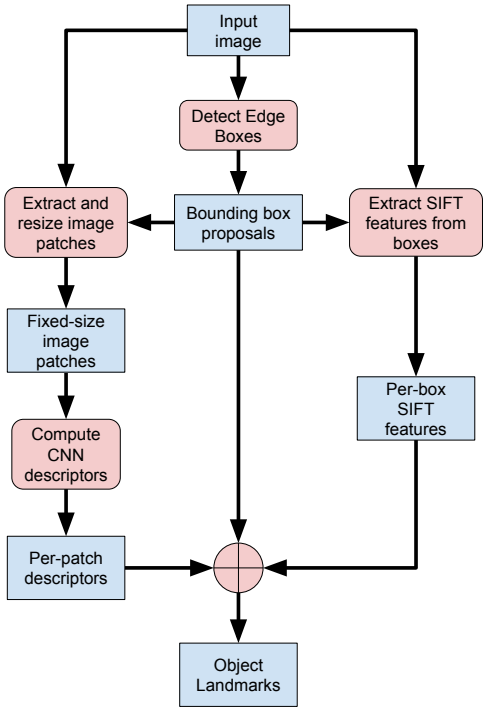


Fig. 1: A schematic of the extraction process for object landmarks. Blue boxes represent data, red boxes represent operations on data. First, Edge Boxes are computed for the image. The image patches enclosed by these boxes are deformed to a fixed shape and passed through a CNN to get descriptors for each object, and SIFT features are extracted from the (original, un-deformed) patches as well. The Object Landmark feature consists of the bounding box, the CNN descriptor, and the SIFT features.

of Zitnick et al. [24], as we found it gave small improvements in accuracy and speed over selective search. Preliminary experiments were also performed with a learned region-proposal network like that used in the Faster-RCNN of Ren et al. [25], but results were very poor by comparison with Selective Search and Edge Boxes, so we disregarded this approach. Parameters $\alpha = 0.55, \beta = 0.55$ were used for Edge Boxes in all experiments. As in [5], we use SIFT for the landmark sub-features in our experiments, because of their well-demonstrated robustness to changes in scale. We use the recommended SIFT configuration proposed in [1]: 3 octave layers, $\sigma = 1.6$ for the initial Gaussian blurring, contrast threshold 0.04, and edge threshold 10.

B. Visual Odometry with Object Landmarks

Given a pair of images I_i and I_j of a scene, we use object landmarks to estimate the transform between the associated camera poses as follows. After extracting landmarks from both images, the cosine distance d_{kl} is computed between every object descriptor $k \in I_i$ and every one $l \in I_j$. Matches are chosen based on the minimum distance and filtered by cross-checking, so that the matches are all those pairs of landmarks

(k, l) such that $l = \arg\min_{l'} d_{kl'}$ and $k = \arg\min_{k'} d_{k'l}$. Once object landmark matches are established, we perform matching between the sub-features of each pair of matched object landmarks, again using cross-checking, but using the Euclidean distance between the SIFT sub-features instead of the cosine distance. This produces a set of point correspondences. If no sub-feature matches can be found between a pair of landmarks, the pair produces a single point correspondence between the centroids of the pair's bounding boxes. All point correspondences from all matches are then used to estimate an essential matrix E using a Random Sample Consensus (RANSAC) process and the five-point algorithm of Nistér [26]. A set of possible transforms are derived from E, and cheirality checking is performed to eliminate inconsistent transforms, leaving a single transform estimate.

IV. NETWORK ARCHITECTURES

For all experiments, we used the ImageNet-trained networks available from the PyTorch deep learning framework [27]. We consider four families of networks: AlexNet, VGG, ResNet, and DenseNet. For each of these we considered several variants of the architecture, except for AlexNet for which there existed only one variant. In order to constrain the size of the feature vectors, all image patches were deformed to 64×64 pixels before being input to the networks, and we did not consider any network layers with activation size greater than 2^{16} floating-point numbers.

AlexNet and the VGG variants are simple feed-forward CNNs. We use each of AlexNet's convolutional and pooling layers as feature extractors. We number the convolutional layers in the order of their application, "conv1" through "conv5", and the pooling layers are numbered for the convolutional layer that precedes them - thus the layer "pool5" pools the output of layer "conv5". The details of the architecture are described in Krizhevsky et al. [3].

We run experiments with four VGG architectures: VGG11, VGG13, VGG16, and VGG19. These names refer to the number of layers with learned weights they contain, the same naming convention followed by ResNet and DenseNet variants. The architectures are described in detail in Simonyan et al. [28]. For each architecture, we use each pooling layer ("pool1" through "pool5") as a feature extractor, as well as each convolutional layer that feeds directly into a pooling layer (designated "pre_pool1" through "pre_pool5"). The VGG architecture variants with batch-normalization are used in these experiments.

ResNet architectures have residual connections, in which the output of a block of layers is summed with its input to produce the input to the subsequent block. We experiment with three of the ResNet architectures described in He et al. [8]: ResNet50, ResNet101, and ResNet152. Each of these consists of a set of five blocks; the features we use are the outputs of each set of residual blocks that performs one downsampling operation, designated "res2c" through "res5c", as well as the output of the first pooling operation, "pool1".

A DenseNet architecture is composed of a sequence of "dense blocks" connected by "transition blocks". Each dense block is a multi-layer CNN in which the input to layer i is the concatenation, along the channel dimension, of the dense block's input with the outputs of layers 1 to $i-1$. Thus, each layer in the dense block has a direct feed-forward connection to every subsequent layer. Transition blocks consist of batch-normalization, a ReLU activation function, a single convolutional layer with learned weights, and finally average pooling. The full details are provided in [9]. We experiment with four DenseNet architectures: DenseNet121, DenseNet161, DenseNet169, and DenseNet201. The dense blocks are labeled "denseblock1" through "denseblock4", and transition blocks are labeled "transition1" through "transition3".

V. EXPERIMENTAL RESULTS

We perform two sets of evaluations over network architectures and layers. The first evaluation is performed on images captured from a moving car, and emphasizes robustness of features to changes in perspective and visual scale. The second evaluation is performed on webcam images of fixed scenes taken over large spans of time, and emphasizes robustness of features to changes in lighting and seasonal appearance.

A. Scale and Perspective

The KITTI Odometry dataset [29] consists of data gathered during 22 traversals of various urban and suburban environments in the city of Karlsruhe, Germany, by a car equipped with a sensor package. The data includes grayscale and colour stereo image pairs captured at a rapid rate, and the first eleven traversals have precise ground-truth poses for every stereo pair. For our experiments, we subsample the frames from the first eleven traversals by a factor of five to limit the scope of the experiment. For each frame in a subsampled traversal, we estimate the transforms between that frame and the next ten frames in the subsampled traversal using the images from the left-hand colour camera, as these pairs exhibit a very challenging range of differences in perspective and visual scale. Frame pairs in which the true angle between the frames' gaze directions is greater than the camera's horizontal FOV are discarded, as the images from such pairs will generally not view any of the same content.

We perform our scale- and perspective-robustness evaluation on the traversals labelled 01, 06, and 07, consisting of 4,055 frame pairs. These were chosen to comprise about 10% of the total KITTI dataset, and to avoid overlapping trajectories. For each frame, the top 500 proposals from Edge Boxes were used to generate object landmarks, and for each frame pair a camera transform was estimated from the frames' landmarks as described in section III-B. Our error metric is the symmetric error in the estimated translation between the two cameras i and j, which we call the pose error:

$$PE_{ij} = || {}^{j}_{i}t_{\text{est}} - {}^{j}_{i}t_{\text{gt}} || + || {}^{i}_{j}t_{\text{est}} - {}^{i}_{j}t_{\text{gt}} || \quad (1)$$

where ${}^i_j t_{\text{est}}$ and ${}^i_j t_{\text{gt}}$ are the estimated and true positions of camera j in camera i's frame of reference. We take the sum of the errors in both frames because the error in camera i's frame is affected by the estimated orientation of camera i but not that of camera j. Summing the two gives a single metric that penalizes errors in the estimated orientations and positions of both cameras.

The results for each family of architectures are presented in Figure 2. Apart from the earliest layers, where performance is generally poor, there is no consistent difference between the performance of different architectures of the same family, especially in the intermediate layers where performance is best. In all of these comparisons, there is a pattern: the earliest layers perform worst of all, while intermediate layers perform best, and the final layers are slightly worse than the intermediate layers.

B. Appearance

To evaluate robustness of CNN features to appearance change, we make use of the webcam dataset published by Verdie et al. [30], [31]. This dataset consists of images taken by fixed cameras of outdoor scenes at different times over a span of several months. For each scene, the dataset captures a range of seasons and weather and lighting conditions. Example images from the dataset are shown in Figure 3.

Because all images for each scene are taken with the same camera pose, we do not perform pose estimation for our evaluation on this dataset. Instead, we exploit the fact that a correct point correspondence between two images should consist of points at the same location in each image. For each pair of images from a scene, we determine point correspondences as in our other evaluations, and we take the l_2 distance between a corresponding point pair's locations as the error of that correspondence. The error for the image pair is the mean of the errors of the point correspondences:

$$AE_{ij} = \frac{\sum_{(\mathbf{p}_i, \mathbf{p}_j) \in P_{ij}} ||\mathbf{p}_i - \mathbf{p}_j||_2}{|P_{ij}|} \tag{2}$$

Where i and j represent two images from a scene, P_{ij} is the set of point correspondences found between i and j, and $\mathbf{p}_i, \mathbf{p}_j$ are corresponding points in i and j, respectively. This error metric is not directly comparable to that used in the perspective and scale experiments, as described in section V-A. The two metrics have different units (meters versus pixels), and the perspective metric is based on the output of a RANSAC process, so is not affected by outlier matches, while the appearance metric is affected by all matches. However, as we are not directly comparing outputs of different metrics, this is not an obstacle to our analysis.

We perform our evaluation on the full-colour test sets from five of the scenes from the webcam dataset, designated Chamonix, Courbevoie, Frankfurt, Mexico, and St. Louis, for a total of $4,345$ image pairs. For each image, the top 250 proposals from Edge Boxes were used to generate object landmarks. For each network layer under evaluation, we compute the mean error over every image pair in each dataset, and then take the mean of these per-dataset mean errors as the overall metric of the layer's performance. The results are displayed in Figure 4.

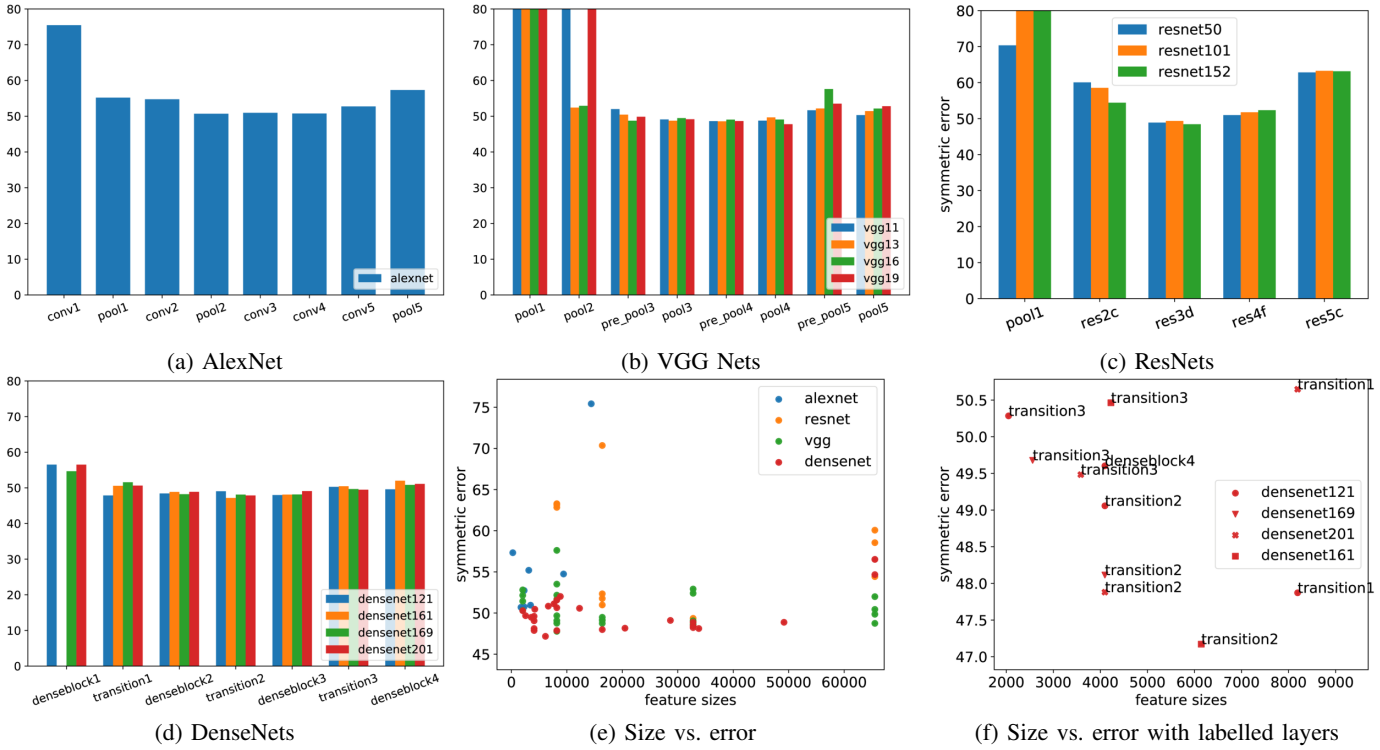
A similar trend is visible in Figure 4 as was in Figure 2: error decreases with the layer's depth in the network up to an intermediate layer, and then increases slightly at the final layers. The trend is smoother here, however: a less sudden drop from the early layers, and more differentiation among the intermediate layers. The discrepancies between network architectures are also more pronounced here than in the KITTI evaluation. AlexNet performs significantly worse at most depths than all other network architectures. ResNets and VGG Nets mostly perform comparably, while DenseNets outperform every other architecture by a considerable margin, with the exception of the "res4f" layers from the ResNets, which have the lowest error of any layer. The DenseNet layers are still favourable because of their greater compactnesss, however: the "res4f" features are almost 300% larger than the "transition3" features of DenseNet161, while the best "res4f" error is only 1.6% lower than DenseNet161's "transition3". Network architecture clearly has a more pronounced effect on robustness to appearance change than scale change.

We note that when applying the same error metric to SIFT features matched alone (not using Object Landmarks), the mean error was 173.09, so the use of Object Landmarks provided a major improvement over plain SIFT regardless of the network layer chosen. The worst-performing layer, "pool1" from VGG11, yielded mean error 105.44, a 64% improvement over plain SIFT. The most compact layer, "pool5" from AlexNet, which at 256 floating-point values is almost an order of magnitude smaller than any other CNN feature considered here, yields mean error 102.28, a 69% improvement. Despite its error being considerably higher than that of many other layers, in some applications it might still be the best choice for its extreme compactness.

Subfigures 4e and 4f plot the errors obtained by different layers against the sizes of the layers. The "transition3" layers again show the best trade-offs of feature size to performance, with the layers from different networks yielding different trade-offs.

VI. DISCUSSION

As a practical takeaway, the results of these experiments most strongly favour the DenseNet family of networks as feature extractors. The best-performing features from this family are as or more robust to changes in perspective as those from any of the other network architectures, while also being among the most compact. Different layers from different architectures within this family provide different trade-offs of robustness and size, but the "transition2" and "transition3" layers of DenseNet161 and DenseNet169 are the most favourable. The "pool5" layer of AlexNet, meanwhile, provides the most compact features by a wide margin, having just 256 floating point values. While their robustness is 16% to 22% worse than most other feature types, their small size and low computational cost may make them a



(a) AlexNet (b) VGG Nets (c) ResNets

(d) DenseNets (e) Size vs. error (f) Size vs. error with labelled layers

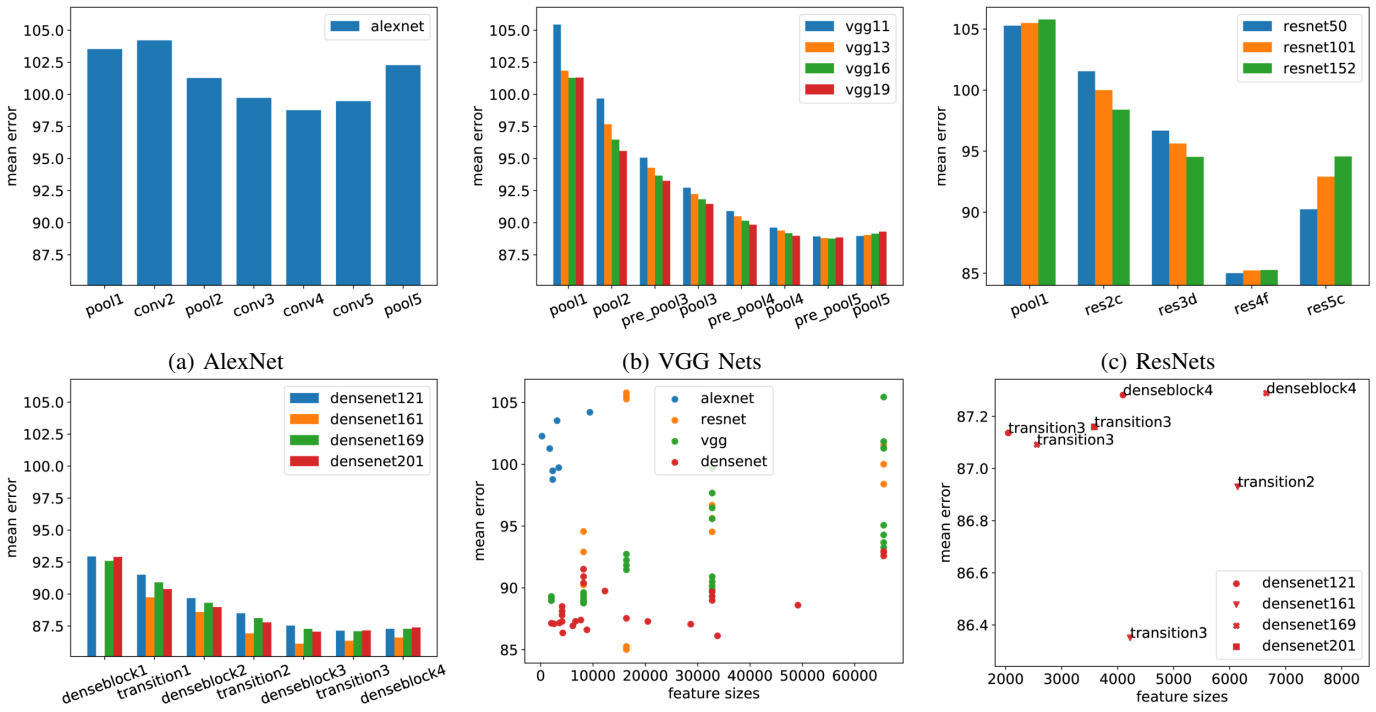
Fig. 2: Subfigures 2a to 2d show mean pose errors, in units of meters, over all image pairs using activations from different layers of different networks as feature descriptors. For sub-figures 2b and 2c, some layers had extremely large mean errors, so the y-axis was scaled such that the heights of the corresponding bars are not visible, in order to make the rest of the plot legible. Note that in Figure 2d, no bar is shown for layer "denseblock1" of "densenet161". This is because this layer exceeded the 2^{16} size limit in that architecture, so no data was collected for it. Sub-figure 2e shows mean pose errors for different descriptors over all image pairs, plotted against the feature sizes of the descriptors. It is apparent that the DenseNet architectures tend to dominate, as they are clustered most closely to the origin. Sub-figure 2f zooms in to show all layers with mean error less than 60 meters and size less than 9000 floating-point values. Only the DenseNet architectures are plotted here.

Fig. 3: Sample images from the Chamonix scene in the webcam dataset.

strong choice in some resource-constrained applications. It does appear, though, that the network architecture makes a much greater difference in robustness to appearance change than to perspective and scale change.

Several interesting trends emerge from the data. One is that the most robust layers of networks tend to be close to the final layer, but are not actually the final layer. This echoes the conclusions of [13] and [17], both of which found intermediate layers to have the best performance. The different networks we analyze here vary widely in their depth, from five learned convolutional layers in AlexNet to two hundred in DenseNet201. The layers we treat as extractors are roughly evenly spaced throughout each network architecture, yet despite their very different depths, the most robust layers are the third-to-last or second-to-last set of layers we evaluate in each network. The performance of a network's layers as feature extractors appears to vary more with the layers' **relative** depth in their network than their **absolute** depth. Our results suggest that the more layers a CNN has, the smaller will be the increment in "semantic richness" provided by each layer. Put another way, with more layers, each layer in the network is allowed to learn a more subtle function of the inputs.

Another noteworthy fact is that the robustness of the features extracted from different networks follows the performance of those networks on the task for which they were trained fairly closely. Table I shows that the ResNet101 and ResNet152 networks used in these experiments both outperform all other networks used on the image classification task for which they were trained, and the "res4f" layers of these networks indeed proved the most robust to appearance change. DenseNets follow just behind, VGGs are worse than that, and AlexNet comes last. This approximately follows the ordering in performance of these network families on the appearance-change task.



(a) AlexNet (b) VGG Nets (c) ResNets

(d) DenseNets (e) Size vs. error (f) Size vs. error with labelled layers

Fig. 4: Subfigures 4a to 4d show mean errors over the webcam dataset across network layers. All plots share the same y-axis, indicating error, to aid in comparing results across plots. Subfigure 4e plots the mean errors against the feature sizes of all layers from each family of networks. From this subfigure it is plain that DenseNets dominate VGG Nets and ResNets, and give much higher accuracy than any AlexNet layer, though some AlexNet layers are more compact. Subfigure 4f zooms in on the lower-left, showing the best-performing DenseNet layers.

Network	Top-1 error	Top-5 error
AlexNet	43.45	20.91
VGG11	30.98	11.37
VGG13	30.07	10.75
VGG16	28.41	9.62
VGG19	27.62	9.12
ResNet50	23.85	7.13
ResNet101	22.63	6.44
ResNet152	**21.69**	**5.94**
DenseNet121	25.35	7.83
DenseNet161	22.35	6.20
DenseNet169	24.00	7.00
DenseNet201	22.80	6.43

TABLE I: Percentage error rates on ImageNet of the pretrained models provided by PyTorch and used in these experiments. Values are taken from PyTorch's online documentation [32]. Note that the documentation does not indicate whether these scores are on the training set or the test set.

VII. CONCLUSIONS

To our knowledge, this work is the most comprehensive evaluation of CNNs as feature extractors to date. Nonetheless, it is not entirely comprehensive - we have not considered ResNeXt or Inception architectures, or any CNNs learned via Neural Architecture Search, which has been shown to achieve superior results to human-engineered CNNs. Future work may expand these experiments to consider more network architectures. It would be interesting to combine this type of analysis with work on neural architecture search, to perhaps gain insight into what elements of network structure have the biggest influence on feature robustness.

In this work, we have compared features extracted from eighty-two layers of twelve different Convolutional Neural Network architectures, evaluating their usefulness in matching tasks under challenging variations in perspective and appearance. We find significant differences both in robustness and feature size among different architectures. Perhaps unsurprisingly, the most recent of the architectures we consider achieves broadly the best performance. We find that the overall best features are the outputs of the third "transition" block of DenseNet architectures, especially DenseNet121 and DenseNet161, which provide slightly different trade-offs of accuracy to feature size. We hope that this comparison will prove useful to others when attempting to choose between many competing network architectures for their own purposes.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523

[2] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, May 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1011139631724

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4] 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 (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[5] A. Holliday and G. Dudek, "Scale-robust localization using general object landmarks," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1688–1694, 2018.

[6] 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*, vol. 1, no. 4, pp. 541–551, 1989. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298594

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[9] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2261–2269.

[10] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2017.634

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00907

[12] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," 2018.

[13] N. Sünderhauf, F. Dayoub, S. Shirazi, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," *CoRR*, vol. abs/1501.04158, 2015. [Online]. Available: http://arxiv.org/abs/1501.04158

[14] N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," in *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[15] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky, "Neural codes for image retrieval," *ArXiv*, vol. abs/1404.1777, 2014.

[16] 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*, ser. CVPRW '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 512–519. [Online]. Available: http://dx.doi.org/10.1109/CVPRW.2014.131

[17] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," *CoRR*, vol. abs/1411.1509, 2014. [Online]. Available: http://arxiv.org/abs/1411.1509

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[19] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 118–126.

[20] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 467–483.

[21] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," 2017.

[22] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images," 2018.

[23] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. [Online]. Available: https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013

[24] L. Zitnick and P. Dollar, "Edge boxes: Locating object proposals from edges," in *ECCV*. European Conference on Computer Vision, September 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/edge-boxes-locating-object-proposals-from-edges/

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Neural Information Processing Systems (NIPS)*, 2015.

[26] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–777, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2004.17

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "Tilde: A temporally invariant learned detector," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[31] N. Jacobs, N. Roman, and R. Pless, "Consistent temporal variations in many outdoor scenes," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

[32] "Torchvision models," https://pytorch.org/docs/stable/torchvision/models.html, accessed: 2019-09-12.