Secrets in Computing Optical Flow by Convolutional Networks

August 16, 2017
Junxuan Li (u5990546)
The Australian National University
u5990546@anu.edu.au

Abstract

Convolutional neural networks (CNNs) have been widely used over many areas in compute vision. Especially in classification. Recently, FlowNet and several works on optical estimation using CNNs shows the potential ability of CNNs in doing per-pixel regression. We proposed several CNNs network architectures that can estimate optical flow, and fully unveiled the intrinsic different between these structures.

1. Introduction

That motion shown in two adjacent images or frames in an video is called optical flow. Optical flow estimation is the algorithms to estimating motion between two images or frames. The most general version of optical flow is to compute the independent of motion at each pixel of an image.

As recently, the convolutional neural networks have been widely used in many areas of computer vision. We want estimate the optical flow by introducing a network architecture without any full connective layers and downsampling. Because we have a intuition that if we want to get a output that is as the same size as input, the downsampling of a network maybe unnecessary. However, as the experiments went on, we found that this kind of network performs bad no matter what kind of techniques were used in training until the downsampling layers were introduced. Several experiments and researches were done in order to gave a reasonable explanation and unveil the intrinsic problems inside this phenomenon.

Finally, we proposed a deeper network with downsampling and upsampling layers that works quite well in this problem. This new network will take two resources as input. One is the raw images pairs, the other is an approximately flow that derive from some other quick methods. We want this CNNs can both fine tuning the approximately flow and take it as guide to compute a more accurately flow.

2. Related work

Classical models: Optical flow has been analysis well over past 30 years. The most widely used models that defines the problem in optical flow is the brightness constancy and spatial smoothness assumption introduce by Horn and Schunck [10]. Many other method make some extract assumption base on these two models. Such as classical++ [20] apply the median filter together with this model to eliminate the outliers in the quadratic formulation. EpicFlow [19] introduce edge-preserving matches so that this model would be more robust to large displacement. However, all the method base on this model can not deal with all the constraint of optical flow in realistic scenes. Such as brightness change that are conducted by changing of light field rather than the motion of the object. These methods are all base on constraints and model that design by human, not any learning from realistic dataset. The progress of these kinds of method will demand more and more human design patterns as long as people want these models to fit the realistic better. It seems that a automatic learning way to do these pattern design is valuable which may highly decrease the human design patterns among the estimation. Also, these methods are usually slow.

CNNs: Now the CNNs is a new technology that performs well on most of the areas in computer vision such as classification [15] [13] and recognition [9]. People found out that the convolution network is very suitable in many computer vision areas, for it can extract high-level feature maps from each images over large spaces. Especially as the growth of computability in GPU and size of dataset we can get from Internet, learning a deep complex neural network become more and more computation feasible. But nowadays, most of the task that CNNs do is classification. It also shown its ability in per-pixel regression such as semantic segmentation [16] and depth estimation [8]. We want to discover ability that CNNs have in per-pixel regression tasks.

CNNs on optical flow: Recently, there are many works that introduce some different CNNs in doing optical flow computation. FlowNet [7] is the first one trying CNNs on

optical flow. And its network architecture is widely used by other researches [11] [18]. This paper not only provide a thought in how to build a network for optical flow, but also generated a big dataset that can be used in training optical flow, the FlyingChairs. But the shortage among this paper is also obvious. As the FlyingChairs is generated by human and its structure of motion is rather simple, only contains rigid motion. So the performance of this network is highly restricted by training dataset, as it will overfitting on training data. It can be seen that on it performs far more better on testing data in FlyingChairs. But not comparable with many other start-of-art methods in other dataset.

Unsupervised CNNs: Many method use unsupervised CNNs [12] [1] [24]. [24] only change the ground truth in FlowNet to be the results of state-of-the-art classical estimators. So that they can apply this learning to some realistic dataset and not need to be restricted by the ground truth of a dataset. But this method highly relies on the accuracy of state of art method. [12] using brightness constancy and motion smoothness as the loss function of the network rather than a ground truth flow. But the performance is also been restricted by the goodness of the model and assumption. It can not cover all the situations in real life. But these two unsupervised learning indeed get rid of the constraint of dataset. As we can get more data from real life rather than random generate from a single data. It can prevent overfitting to a special situation.

3. First trial on Network architectures

In our opinions, the network [13] would like to use layers with downsampling, by which they can shrink the features over whole image, in order to use a softmax or full connective layer to do the classification. But if our tasks is to do a pixel-to-pixel estimation, the downsampling layer maybe redundant, since the most of them [7] [24] using downsampling layers have to recover the original resolution by several upconvolution layers.

3.1. Network without downsampling

So we proposed a network without downsampling. It is designed as shown in Figure 1. For each convolution block it has, it only contains convolution with 5 by 5 filter window size, batch normalization and RELU, all with 1 pixel stride. For each layers, the input of it is labelled in the figure. Typically, the input of first layer is $H \times W \times 8$ where H, W is the height and width of original images. As the limited of computation of GPU, we could not make this network any bigger. To train this network, we need to store at least $H \times W \times 906$ (combine the input of each layer) values in forward and backward of the network. But only with 22k parameters (906 convolution filters with 5 by 5 windows) to train. So in total, this network should need less data to train, as the parameters is less, and faster to test, as the layer is shallow.

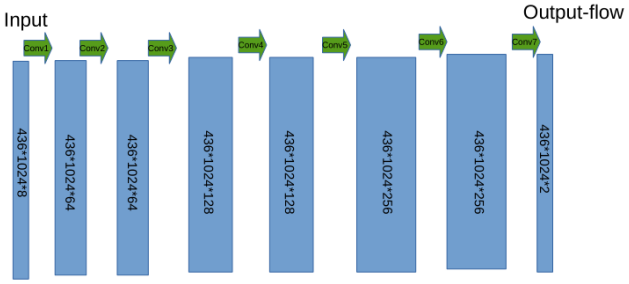

Figure 1: A small network without any downsampling. The green arrow with text `Conv` is Convolutional block with 5 by 5 filter size. One convolutional block will include a convolution and batch normalization and RELU. The number in each block is the input size of each block. e.g. $436 \times 1024 \times 8$ is the input size of first block.

3.2. Error function

As training loss we use the endpoint error (EPE) [2], which is the standard error measure for optical flow estimation.

The normalised endpoint error (NE) between an estimation flow $I(x, y)$ and a ground-truth flow $I_{GT}(x, y)$ is given by:

$$NE = \left[\sum_{(x,y)} \frac{(I(x,y) - I_{GT}(x,y))^2}{\|\nabla I_{GT}(x,y)\|^2 + \epsilon} \right]^{\frac{1}{2}} \tag{1}$$

For CNNs is using back-propagation to update the parameter. We need to derive its partial derivative over each pixel $I(x_i, y_i)$:

$$\frac{I(x_i, y_i) - I_{GT}(x, y)}{[(\|\nabla I_{GT}(x,y)\|^2 + \epsilon) \sum_{(x,y)} (I(x,y) - I_{GT}(x,y))^2]^{\frac{1}{2}}} \tag{2}$$

3.3. Intrinsic problem in this network

However, the performance on this network is bad. The comparison of its results will be shown in Results section. In general, we will easily overfit this network so that it will perform quite well on the training dataset while bad on the testing dataset. Besides, the output of this network seems to be very sensitive to the colour-map of input. But as we know that optical flow should be derived based on relative displacements between frames rather than colour.

So we can conclude that this network can only learning some basic features of an images, such as edges, colour and brightness. It can not extract the high-level features that can be used to compute optical flow. What kind of problem

will rise situation like this? First of all, the training dataset should be enough, as we only have few parameters to train. Also, the performance in the validation dataset show us that it is never getting better till the end of training.

Can the network get any deeper, so that we can extract and process more features than before? No, we are not using any downsampling here, so the memory demanding of this network will increase rapidly after 4 convolution layers and 6 convolution layer is the most layer that we could try temporary in a 11GB GPU. Even if we can get a bigger GPU to train it, the time and cost to train this big network is out of our consideration.

According to Zeiler *et al*. [23], they explored how discriminative the features are in different layers. As the feature hierarchies become deeper, a better performance they can get using a linear SVM to do the classification problem. This shows that neural networks can learn increasing powerful features over layers. They also build a way to reconstruct the high-level features stored in low-dimensional space by an operation called unpooling. It seems that as the layer being deeper, the features it generate is more specific and discriminative. Usually the first 3 layers are edge and colour features which are very basic and not informative. But as the pooling and convolution get this network deeper and smaller. The features that been extracted show high discriminative of an object and even if the resolution of a feature is small, they can also construct the intact information in original images. These unpooling and deconvolution techniques are also been used in several papers doing semantic segmentation [16] [17] which is also a task doing pixel-to-pixel estimation. By doing this, they can extract high-level features from original images over and large areas by pooling or downsampling layers. An output of original resolution can also been reconstruct by concat information from high-resolution with informative but low-resolution features. Been inspired by all the meaningful of doing downsampling, we proposed and new network architecture that can both make used of high-level features and reduce the computation while upsampling.

4. Final Network architecture

We design an network shown in Figure 2. All convolutions in our network is using a 1 pixel stride and all pooling is with 2 pixels stride, all with appropriate padding. When a convolution block is apply to a layer, it will do a series action, first convolution, then batch normalization, then RELU, and may contains a pooling operation, which depends on whether or not we will do a downsampling in this layer. As we do downsampling over the layers, the output is thus at half the resolution of input resolution. After four convolution blocks with downsampling, we will get $1/16$ of the original size of input. Apparently, this kind of size could not be output as a dense optical flow. So we design a bilinear upsampling layer which can to obtain an output twice big as the input. A flow field of original resolution can be obtain after several bilinear upsampling layer is applied.

4.1. Network input

The input of the network is a $H \times W \times 8$ space of pixels made by combining two adjacent RGB images and its approximate flow. First six channels of input is the RGB value of two frames, the other two channels in the end is the uv map of approximate flow. The reason we also want to take an approximate flow as input is that if we only train this network with RGB channel, the network can not learn much from the dataset. The performance is not as good as a approximate flow. So we want to add this to guide the network go for a better training. And this approximate flow we add indeed improve the performance of the network, so that we can make this it work even with a small training dataset.

This approximate flow is only used as a guided flow so that network can estimation the final flow easily, it can highly decrease the demanding of training data while not make much influence of flow accuracy. For this approximate flow is just a pre-computation of the input, it should not be time consuming. In fact, we want this pre-computation as fast as possible while not losing the general information of a coarse optical flow. Because one of the advantages of CNNs is the speed on running time. We want to make the this optical flow estimate outperform other classical methods in run times.

There are several methods that perform well in optical estimation such as EpicFlow [19] and LDOF [4]. But both of them take more than 10s to compute a frame. DIS-Fast [14] and PCA-Flow [22] is much faster than previously two, they take around 0.1 seconds to do the estimation. These two could be a good method to do the approximate flow computation. However the codes to implement these two method is not released. I try to use Block matching [3], a patch based method which could do a coarse motion estimation.

4.2. Bilinear upsampling

Applying pooling layers or convolutions with stride two will make CNNs good at extracting high-level features that human unreadable from images. Besides, downsampling is necessary to make network training computationally feasible. The networks without any pooling or convolution with 2 pixel stride will not only demands a huge computation ability of computer, but also can not aggregate the interesting features over large areas of big feature maps. However, pooling will cause output layer have a half resolution of input feature maps, so in order to compute dense flow in each pixel, we need a refinement in those sparse feature maps to get results of original resolution.

In 'FlowNet Simple' by Dosovitskiy *et al*. [7], they used

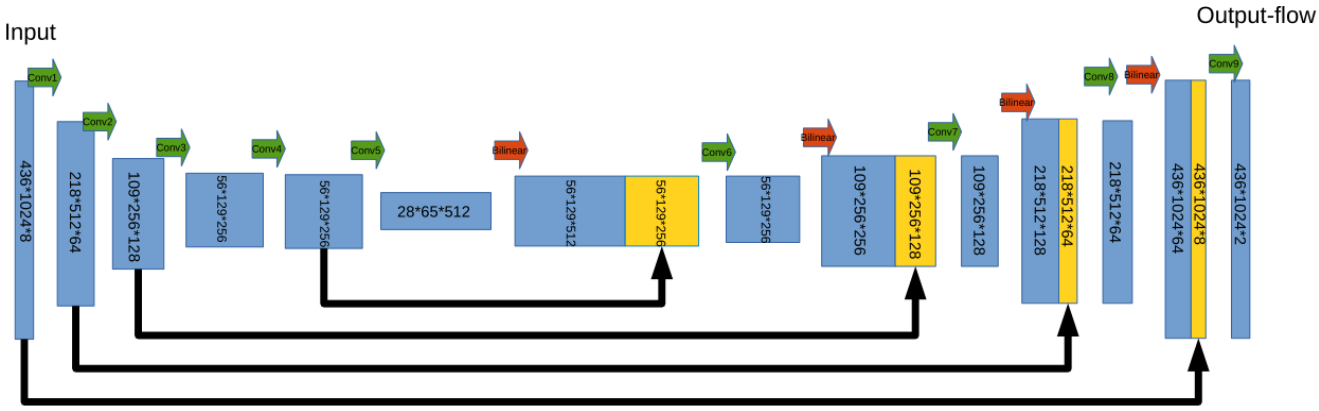

Figure 2: Final version of our network. The green arrow with text `Conv` is Convolutional block, it only contains pooling when it need to do the downsampling. The third dimension of feature maps increases as the network go deeper, it is doubling after each convolutional block with max pooling stride two. The red arrow with text `Bilinear` is the bilinear upsampling block. It will simply bilinear resize the input to a twice big space. The black arrow from previously layers to post layers is the concat action which simply concatenate two layer in third dimension.

a 'upconvolutional' layers, consisting of unpooling (extending the feature maps, as opposed to pooling) and a convolution. While we used a computationally less expensive bilinear upsampling. To perform the upsampling and continue with feature dimensional shrinking, we apply the 'Bilinear upsampling' to a small feature maps, and concatenate its output with suitable size of previous feature maps in the network and an apply convolution block after concatenation. By doing this we can preserve the dense information passed from previous feature maps together with the high level information over large areas that extracted by pooling operations. Each a bilinear upsampling and a convolution will increase the resolution twice and shrink the third dimension of a feature maps to one third. We repeat it 4 times. At the last convolution block, we make its output channel 2 to be a prediction layer, resulting in a dense optical flow for which the resolution is the same as original images.

5. Implementation

5.1. Training dataset

Unlike traditional approaches such as [6] only learning a few parameters, neural networks have million parameters to train from scratch. Besides, it is also different from some problems in images classification, the dense ground truth optical flow of two images is hard to get. The following dataset is suitable for training and testing.

MPI-Sintel dataset [5] is a large dataset available which contains 1041 training image pairs. It displacement of ground truth flow vary from small magnitude to large, which will make the training go overfitting. **Middlebury** dataset [2] has 8 image pairs with ground truth. Its magnitude of flow usually less than 9 pixels. We use this dataset as the testing dataset.

5.2. Training CNNs

For training CNNs, we use `'MatConvNet'` [21] framework with few changes. We choose standard stochastic gradient descent with momentum as optimisation method. We use a small mini-batches of 6 image pairs. To prevent from overfitting during training, we split the Sintel dataset into 900 training and 141 validation pairs. We start with learning rate $\lambda = 10^{-7}$, and reduce to $\lambda = 10^{-8}$ after half iterations.

6. Results

The training curve of first trial network (denoted as PlainNet) and final network(denoted as FinalNet) is shown in Figure 3. We can see that error of validation on PlainNet does not change much, but the error in training dataset keep reducing, which means that PlainNet learning few things from intrinsic logic of optical. While FinalNet can keep both validation and training reduce until 100 epochs. The FinalNet seems to converge in around 100 epochs, and start a little bit overfitting from 100 to 200 epochs. For the error of training dataset is keep reducing while the error of validation dataset is not changed. So we choose network in epoch 150 to be the final parameters of network.

It can be seen from Table 1 that, for the data that we never met during the training (Sintel Validation and Middlebury Testing) we can get around 8% improvement on FinalNet. But the PlainNet is only working on training dataset, which still shown the overfitting of it.

There are some example results shown in Figure 4. We can see from the images that most of the outliers in block

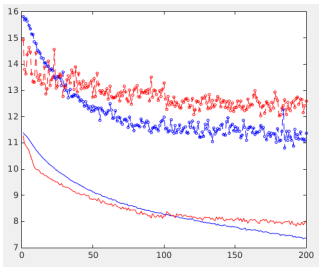

Figure 3: Training curve, blue line is the error of FinalNet, red line is the error of PlainNet. Solid line below is represent for training dataset error, dashdot line is represent for validation dataset error. X coordinate is number of epochs, Y coordinate is endpoint error.

EPE	Block Matching	PlainNet	FinalNet
Training	9.19	8.01	7.13
Validation	12.14	12.61	11.22
Middlebury	3.09	3.61	2.90

Table 1: Average endpoint errors (in pixels) of my networks

Method	Time CPU	Time GPU
EpicFlow [19]	16	-
LDOF Validation [4]	65	2.5
FlowNetS [7]	-	0.08
FlowNet2-s [11]	-	7
Our-PlainNet	0.3	0.16
Our-FinalNet	0.3	0.04

Table 2: Running time compare with other methods

matching results can be eliminate in output flow by FinalNet. And it can also help to correct some deviation that generate in block matching. While PlainNet remains the edges and colour features of the original images, it can not predict flow.

In Table 2 we show the runtime of each image pair by different methods. Although our method is not as good as others in accuracy, we can out perform most of the classical method in runtimes. Besides, both the runtimes and performance of FinalNet outperform PlainNet. Both training and testing of the networks is been done by NVIDIA GTX 1080 GPU while the computation of approximately is on CPU. The runtimes of other method is taken from their papers.

7. Discussions

The most important thing in this network is the downsampling. If we want to get a output that is as the same size as input, why do we want to do downsampling and upsampling in the network? Is all convolution without any stride two a good design? The answer is clearly no after we tried an network without any downsampling. For PlainNet will not shrink the features over layers, its computation demands much more than FinalNet. Besides, it seems that PlainNet will overfit much. The output of PlainNet is very sensitive to the color of a image rather than displacement. We think the reason would be that downsampling can help network to extract useful features. And as the network get deeper, it can prevent overfitting.

8. Conclusions

In summary, we show how important the downsampling layers are in a network. We also have proposed a new network that can do pixel-to-pixel estimation by combining high-level features with images details. Our approach can also combine with some other state-of-art optical flow estimation methods in the future, and see how it work with them.

References

[1] A. Ahmadi and I. Patras. Unsupervised convolutional neural networks for motion estimation. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 1629–1633. IEEE, 2016.

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[3] A. Barjatya. Block matching algorithms for motion estimation. *IEEE Transactions Evolution Computation*, 8(3):225–239, 2004.

[4] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 41–48. IEEE, 2009.

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[6] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2443–2450, 2013.

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[8] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[9] S. Hijazi, R. Kumar, and C. Rowen. Using convolutional neural networks for image recognition, 2015.

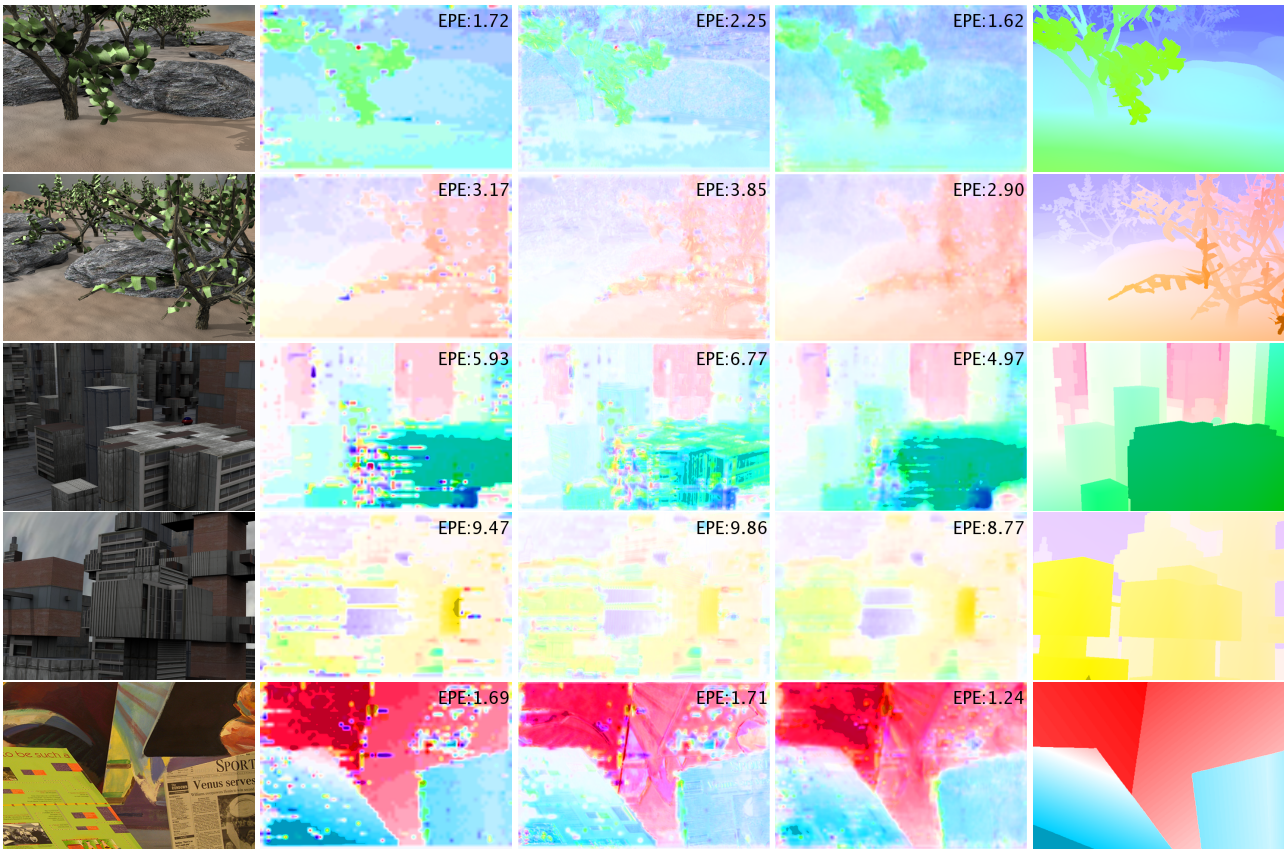

Figure 4: Examples of optical flow prediction on the Middlebury dataset by our network. In each row left to right: original image, block matching results, output of PlainNet, output of FinalNet, ground truth flow. Endpoint error is shown for every frame. Note that even though the EPE of block matching results is usually worse than that of output of network, the networks can refine the flow and make it more accurate.

[10] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[11] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[12] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Computer Vision–ECCV 2016 Workshops*, pages 3–10. Springer, 2016.

[13] 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.

[14] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016.

[15] 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*, 1(4):541–551, 1989.

[16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[17] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[18] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016.

[19] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[20] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[21] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[22] J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 120–130, 2015.

[23] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[24] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Guided optical flow learning. *arXiv preprint arXiv:1702.02295*, 2017.