# Optical Flow Requires Multiple Strategies (but only one network)

Tal Schuster[1] Lior Wolf[1,2] David Gadot[1]
[1]The Blavatnik School of Computer Science, Tel Aviv University, Israel
[2]Facebook AI Research
talschuster@gmail.com, wolf@cs.tau.ac.il, dedigadot@gmail.com

## Abstract

*We show that the matching problem that underlies optical flow requires multiple strategies, depending on the amount of image motion and other factors. We then study the implications of this observation on training a deep neural network for representing image patches in the context of descriptor based optical flow. We propose a metric learning method, which selects suitable negative samples based on the nature of the true match. This type of training produces a network that displays multiple strategies depending on the input and leads to state of the art results on the KITTI 2012 and KITTI 2015 optical flow benchmarks.*

## 1. Introduction

In many AI challenges, including perception and planning, one specific problem requires multiple strategies. In the computer vision literature, this topic has gained little attention. Since a single model is typically trained, the conventional view is that of a unified, albeit complex, solution that captures all scenarios. Our work shows that careful consideration of the multifaceted nature of optical flow leads to a clear improvement in performing this task.

In optical flow, one can roughly separate between the small- and the large-displacement scenarios, and train model to apply different strategies to these different cases. The small displacement scenarios are characterized by relatively small appearance changes and require patch descriptors that can capture minute differences in appearance. The large displacement scenarios, on the other hand, require much more invariance in the matching process.

State of the art methods in optical flow employ metric learning in order to learn the patch descriptors. We focus on the process of selecting negative samples during training and suggest two modifications. First, rather than selecting all negative samples close to the ground truth, we propose an *interleaving learning* method that selects negative samples at a distance that match the amount of displacement that the true match (the positive sample) undergoes, as is illustrated in Fig. 1. Second, we suggest gradually increasing the difficulty of the negative samples during training.

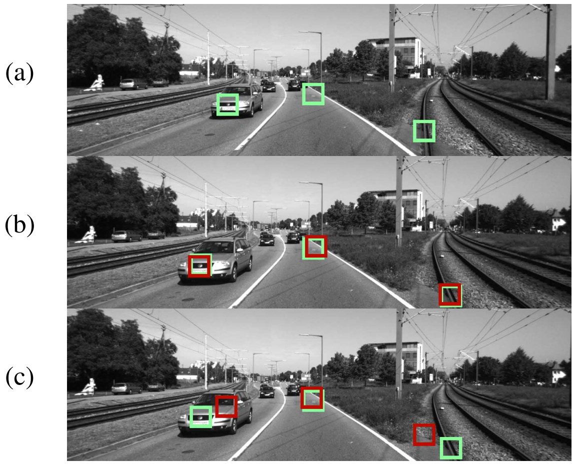

Figure 1. Illustration of strategies for selecting negative samples. (a) The first frame, in which some locations are marked. (b) In the baseline method [13], negative samples are sampled close to the ground truth, regardless of the properties of the true match. Green are the true matches and Red are the negative samples. (c) In the proposed method, the negative samples are sampled based on the displacement of the positive samples. Best viewed in color.

In the implementation of the second component, scheduling samples by difficulty, we combine two methods well known in the literature. The *curriculum learning* method [6] selects samples, stratified by difficulty, using a predefined order. The method of *self-paced learning* [24] identifies a set of easy samples by their loss, and learns using only those samples. The amount of samples defined as easy is increased over time. The *Self-Paced-Curriculum-Interleaving* method we propose here combines in the selection process both the difficulty of a sample and its loss. However, in difference from the *self-paced* method, no samples are excluded during training. Instead, we control the level of the difficulty of instances used for training by selecting negative samples of appropriate distances.

The pipeline employed for computing optical flow is

similar to the PatchBatch method [13]. We slightly modify it by replacing the DrLIM loss with a Hinge loss.
Our main contributions in this work are:

- We analyze, for the first time, the need for multiple strategies in optical flow.
- We propose a novel, psychologically inspired way to train a network to address multiple scenarios at once.
- We show how, in optical flow, our proposed new scheme translates to a simple, unexpected, heuristic.
- We improve the PatchBatch[13] pipeline itself.
- State of the art results are demonstrated on the KITTI 2012 and KITTI 2015 benchmarks.

## 2. Related work

Many computer vision tasks require a pixel-wise image comparison (*e.g*. image retrieval, object recognition, multi-view reconstruction). To allow for the comparison to be invariant to scale, rotation, illumination, *etc*., image descriptors such as SIFT [28], SURF [5], HOG [10], and DAISY [35] have been used. Brox and Malik were the first to apply local descriptors to the problem of dense optical flow [7]. They found that the use of descriptors enables better performance for large displacement matching, but that the obtained solution has many outliers due to missing regularization constraints. In order to account for this, they used descriptors to build a sparse initial flow and interpolate it to a dense one using image smoothness assumptions. Following their success, many other models adopted the use of local descriptors [39, 30, 20, 34].

With the advent of deep learning methods, CNNs were shown to be extremely powerful in the related problem of stereo matching [33, 41]. For optical flow, a few CNN based models were proposed. In [37], a CNN is used to predict the flow from a single static image. FlowNet [11] is the first end-to-end CNN for optical flow and showed competitive results. In the PatchBatch [13] pipeline, a CNN was used for extracting patch descriptors that are then used for matching via the PatchMatch [4] Nearest Neighbor Field (NNF) algorithms. It achieved state of the art performance in the KITTI benchmarks [15, 29] as of last year.

While the use of descriptors has greatly improved overall performance and accuracy, methods keep failing with large displacements, as we further discuss in Section 4. To solve this problem, extensive efforts have been devoted to methods for the integration of descriptors with local assumptions [7, 34, 30]. However, much less work was done in making the descriptors themselves more suitable for this scenario. A concurrent work [3], focused on decreasing the error for large displacements by down-sampling patches and adding a threshold to the loss function. However, this comes at the cost of reducing the accuracy obtained for small displacements.

In our work, we follow the PatchBatch pipeline and use a CNN to extract descriptors. We expand the work by analyzing different matching cases, specifically those of small and large displacements, and present a method for generating better matching descriptors for both cases.

### 2.1. Learning for multiple strategies

The need for multiple strategies was found in several vision problems where the basic trained model could not optimize the solution for all sub-categories. An example is the work of Antipov *et al*. [1] for age estimation. Unsatisfied by the accuracy of the model for children of age 0-12, they train a sub-model only for those ages and employ it to samples that are classified as this category by another model that is run first.

Another common case is in fine-grained classification, *e.g*. determining the exact model of a car or a particular species of bird. The subtle differences between nearby species require, for example, to focus on specific body regions. However, different distinctions require different body parts and we can consider each body part as a separate decision strategy.

In order to achieve the required accuracy, some methods perform object segmentation [23] or part detection [22] to limit the search of each sub-class to the most relevant body parts. A different approach was shown in [14], where several models were trained on different samples to create per class expert models. At test time, the answer with the highest confidence is chosen. The latter approach achieved better results due to each model leveraging all of the input data, and learning individually the required features to gain expertise in its task.

### 2.2. Learning for varied difficulty levels

*Curriculum learning* [6], inspired by the learning process of humans, was the first method to manipulate the order of samples shown to the model during training. Specifically, it is suggested to present the easy training samples first and the harder samples later, after performing stratification based on the difficulty level.

In *self-paced learning* [24], instead of using a predefined order, the difficulty of each sample is dynamically estimated during training by inspecting the associated loss. On each epoch, only the easier samples are being learned from and their amount is increased with time until the entire data is considered. In the work of [19], those two methods were combined to allow a prior knowledge of samples difficulty to be considered in the *self-paced* iterations.

It was recently proposed to eliminate from the training process samples that are either too easy or too hard [36]. For this purpose, specific percentiles on the loss were employed.

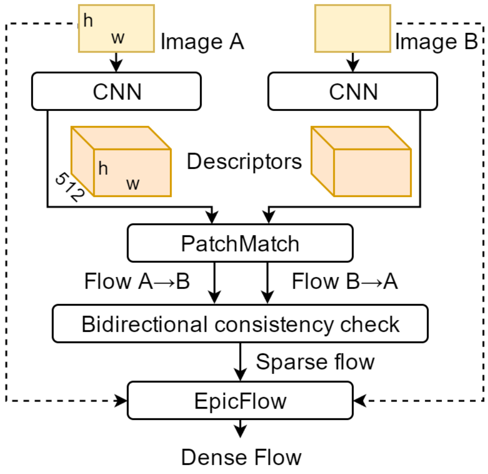

Figure 2. Flow diagram of the PatchBatch pipeline. The same CNN is applied for patches from both images. PatchMatch [4] is applied twice in order to get both flow directions.

Samples which did not meet the loss criteria were put aside for a predefined number of epochs.

In the problem of optical flow, large displacements are known to be more challenging. Moreover, as we show in Section 4, the descriptor extraction strategy should differ by displacement. Due to the correlation between the difficulty level and the required strategy, applying the existing gradual learning methods could result in acquiring specific strategies in different training stages with the possibility of unwanted carryover. In Section 5, we suggest novel learning techniques, which use all samples, support different strategies and apply an easy to hard order.

## 3. The PatchBatch pipeline

The PatchBatch (PB) pipeline, as described in Fig. 2, consists of a CNN which generates per-pixel descriptors and an approximate nearest neighbor algorithm which is later used to compute the actual assignments. PatchBatch's ACCURATE network configuration generates descriptors with 512 float values. The assignment is computed by minimizing the $L_2$ distance between descriptor vectors. To create each pixel's descriptor, the CNN uses a patch as an input. In most of the CNN configurations described in PatchBatch, the input is a $51 \times 51$ patch centered around the examined pixel. The CNN uses the grayscale data of the patch to extract a descriptor as similar as possible to the one extracted for the matching pixel on the second image.

Using the generated descriptors, PatchMatch [4] (PM) algorithm is used to compute initial flow assignments. PM is applied in both flow directions and is followed by a bidirectional consistency check that allows elimination of non-consistent matches.

In the final step, the sparse-to-dense EpicFlow [32] (EF) algorithm creates the final estimation using the sparse flow and the original raw images. We refer the reader to the PatchBatch [13] paper and the published code[1] for a more detailed description.

### 3.1. Architecture improvements

In this paper, we improve the CNN that generates the descriptors. We achieve this by several means. First, we adopt the suggestion, that was partially tested in the original PB paper [13], to enlarge the patch size from $51 \times 51$ to $71 \times 71$ pixels. Second, to improve the training of the network we use two novelties: (1) We introduce a new learning method for multiple displacements detailed in Section 5. (2) We modify the loss function and use a new form of the Hinge loss. Third, we altered the initial random guess range of the PM algorithm on MPI-Sintel to be 100 instead of 10, to allow larger search distance and better utilization of our large displacements descriptors. For the KITTI benchmarks, this parameter remained unchanged (500).

### 3.2. Hinge loss with SD

Instead of the DrLIM [17] loss functions used in PatchBatch, we found the Hinge loss to achieve best results when integrated with our, further detailed, learning method. To allow the use of this loss, we construct the samples as triplets. For each patch, we collect a matching patch by the ground truth and a non-matching one. As a baseline, we use the same non-matching collecting method, which is a random patch up to 8 pixels from the matching one.

We define the loss function as:

$$L_H = \frac{1}{n}\sum_{i=1}^{n} max(0, m + D_{i,match} - D_{i,non-match}) \tag{1}$$

where D is the $L_2$ distance of the examined patch descriptor from the matching or non-matching one.

In the PatchBatch paper, an addition of a standard deviation parameter was found to produce better distinction between matching and non-matching samples. With that inspiration, we apply a similar addition to the Hinge loss:

$$L_{H+SD} = \lambda L_H + (1-\lambda)(\sigma_{D_{match}} + \sigma_{D_{non-match}}) \tag{2}$$

We used $m = 100$, $\lambda = 0.8$ and a training set of $n = 50k$ triplets for each epoch.

## 4. Optical flow as a multifaceted problem

It is clear by examining the results of the common optical flow benchmarks that optical flow methods are challenged by large displacements. In the MPI-Sintel [8], where results are separated by the velocity of pixels, the current average

[1] https://github.com/DediGadot/PatchBatch

| Train set | 0-5 | 5-10 | 10-20 | 20-30 | 30-45 | 45-60 | 60-90 | 90-$\infty$ |
|---|---|---|---|---|---|---|---|---|
| Baseline (all) | **2.32** | 7.32 | 5.32 | 9.38 | 25.21 | 50.43 | 67.32 | 216.39 |
| <30 | 2.46 | **6.91** | **5.25** | **8.57** | 26.39 | 51.76 | 65.15 | 209.40 |
| >30 | 3.03 | 9.07 | 5.64 | 10.29 | **24.74** | **46.81** | **56.69** | **199.61** |

Table 1. The increase of distractors with displacement and the success of models trained on a partial range, shown as average distractors amount by displacement range. The number of distractors for a given patch is the number of patches whose descriptors are within a smaller distance from it than the true match. Each column show the results for the Hinge+SD PB model trained on a specific displacement range.

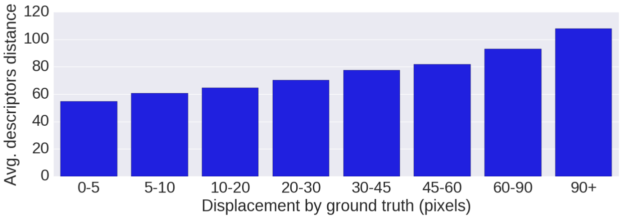

Figure 3. Correlation of larger displacements with larger distances between the true matches. The average $L_2$ distance between descriptors of matching patches are shown grouped by displacement range. The descriptors were generated using a trained Hinge+SD PB model on the KITTI2012 benchmark.

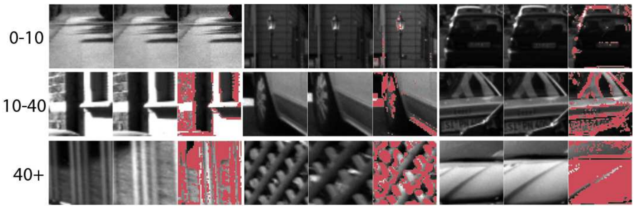

Figure 4. Extent of pixel displacement is correlated with apparent differences in the KITTI benchmark. Samples are gathered in triplets, in which the matching pair is next to a display that depicts by red dots locations with $L_1$ distance larger than 0.2 between pixels value. Each row show examples from a displacement range that appears to the left. Best viewed in color.

end-point-error (EPE) of the top 10 ranked methods is 35.47 for velocities higher than 40, while it is about 1.01 for velocities lower than 10. In KITTI2015 [29], there is no published estimation by velocity. However, there is separation of foreground vs. background regions. The current average outliers percentage for the top 10 methods is 26.43% for foreground versus 11.43% for background, which, assuming foreground objects typically move faster than background, supports the same observation. When evaluating the baseline PatchBatch model on a validating set, we notice an error (percent of pixels with euclidean error $>$ 3) of 4.90% for displacements smaller than 10 and 42.15% for displacements larger than 40.

The challenge of matching at larger distances is exemplified in Fig. 3, which shows the $L_2$ distance of the true match as a function of the ground truth displacement. Furthermore, as the distance increases, the average number of distractors in the second image, with higher similarity to patch in the first image than the true match, increases. This counting is performed in a radius of 25 pixels around the true match and is shown in Tab. 1 under the Baseline training set.

### 4.1. Multiple strategies

When training the PatchBatch network only on displacements that are smaller than 30, we are able to improve most cases of small displacements, while, in most cases increasing the number of nearby distractors for large displacements. Conversely, training only on displacements larger than 30 pixels, achieved a lower amount of distractors for large displacements (Tab. 1). However, since there is no mechanism for selecting between the two networks, it is best to train one network that addresses both scenarios. Interestingly, when training just one network on all samples, the network seems to outperform the two specialized networks in the domain of very small displacements. This is probably a result of designing the PatchBatch method to excel in benchmarks that emphasize this category.

Large displacements are typically associated with larger differences in appearance, as demonstrated in Fig.4. Differences in the patch appearance for the small displacement case typically arise from objects moving within the patch faster than the middle pixel. In contrast, in large motions, we can expect much more pronounced changes in appearance due to the following: (1) As fast objects move, their background is more likely to change. (2) The view point changes more drastically, which leads to different object parts being occluded. (3) The distance and angle to light sources vary more quickly, leading to a change in illumination. (4) When a significant displacement occurs along the Z-axis of the camera, the object changes in both position and scale.

## 5. Learning for multiple strategies and varying difficulty

As baseline methods, we apply gradual learning methods from the literature. For applying curriculum learning [6], the samples need to be stratified by difficulty prior to training. Followed our previous findings, we define the difficulty level as the displacement value in the ground truth and increase the maximum displacement of the sample pool in each epoch which we call *curriculum by displacement*.

Another curriculum implementation, which we call *cur-*

*riculum by distance*, would be to use samples with all displacement values for each epoch, and to start the training using false samples that have a large euclidean distance in the image from the true matching. Decreasing that distance with training should provide harder false samples with time.

We also implement a self-paced model by learning only from the easy samples in each epoch. Easiness here is measured per sample by requiring a loss that is lower than a threshold. The threshold increases over the training.

### 5.1. Interleaving learning

We present a novel learning method for machine learning, motivated by the cognitive literature.
Both the curriculum learning approach as well as the self-paced one utilize the difficulty diversification of the samples and suggest to learn from easy to hard. While this idea might seem appealing, and does work in many machine learning problems, it could cause the network to become overly adapted to different aspects of the problem at different training stages. In optical flow, models must excel in the low displacement task in order to be competitive. Therefore, the shift of attention to harder and harder tasks is potentially detrimental. In addition, if different strategies are required, the carryover from the easy task to the more challenging ones is not obvious.

Our approach is motivated by psychological research. Kornell and Bjork, psychology researchers, found that for some cases, interleaving exemplars of different categories can enhance inductive learning [21]. Their tests showed that people learn better to distinguish classes, *e.g.* bird species, by learning in an interleaving sample order rather than blocks of the same class. Another example would be sports training, in which it is common to interleave simple basic exercises with more complex ones, incorporating at least part of the complex movements from very early, and going back to the basic movements even after these are mastered.

The idiomatic way of training ML models is to randomize the feeding order of the samples. When perceptual strategies and difficulty levels are unrelated, the random process might be sufficient. However, when the samples that require some strategy A are consistently harder than the ones required for strategy B, the frequent loss related to the samples associated with A would mean that the strategy B would be deprived of a training signal.

To preserve a random order of strategies, and, at the same time, facilitate the penalty of harder samples, we suggest that the learning process should consider the difficulty of each sample. This could be done by either taking the difficulty of the sample into account while computing the penalty or, when training by pairs or triplets of samples, by controlling the composition of these small reference groups.

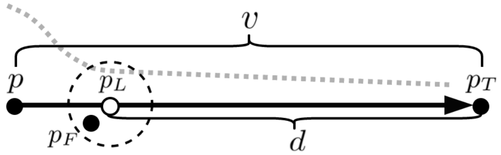

Figure 5. Illustration of the false sample collecting methodology for interleaving learning. $p$ and $p_T$ represent a location in the first frame and its true matching from the second frame respectively. $p_L$ is sampled along the motion line ($p \rightarrow p_T$). The false sample ($p_F$) is randomly chosen from inside the dashed area that is 8 pixels from $p_L$. The dotted gray line represents the log-normal distribution from which $p_L$ is taken (mostly closer to $p$).

### 5.2. Interleaving learning for optical flow

The implementation of our method was done by using further patches as false samples for larger displacements. Thus, for the harder case of large displacements, we select false samples that should be easier to distinguish from the true ones and normalize the overall difficulty. From the strategy point of view, by presenting further away negatives for large displacements, the model learns to rely more on context and less on appearance changes for large displacements and conversely for small ones.

The chosen false sample distance is determined by:

$$d = v(1 - X) \qquad X \sim \log\mathcal{N}(\mu, \sigma) \tag{3}$$

$$P(X = x) = \frac{1}{\sigma x\sqrt{2\pi}} e^{(-\frac{(ln(x)-\mu)^2}{2\sigma^2})} \tag{4}$$

where $v$ is the displacement of the matching pixels and $X$ is sampled from a log-normal distribution [31].

Using a log-normal distribution, allows us to take samples mostly relative to the exemplar motion while also providing a small amount of harder samples. We used $\mu = 0$ and $\sigma = 1$ as parameters and after sampling values for all of the batch samples, they were normalized to $[0, 1]$.

To implement this method in our learning process, we collect the false sample along the line connecting the original and the destined coordinates of the patch. Specifically, we randomly select a sample from a radius of up to 8 pixels from the point with distance $d$ from the true match on that line, in the direction of the position in the first image (see Fig. 5). Interestingly, for the purpose of creating dual strategy descriptors, it does not matter whether the samples are from along the motion line. However, in our experiments, it turned out that sampling this way slightly helps the subsequent PM step. This is probably because PM initially searches in a random distance from the original patch position. By taking a false match that is closer to the original location, we help eliminate those samples.

### 5.3. Self-Paced Curriculum Interleaving learning

Given the interleaving learning method, which, unlike curriculum learning employs all samples at once, we can ex-

| Model / Learning method | Error percent post PM | post EF | Distractors amount by displacement range 0-5 | 5-10 | 10-20 | 20-30 | 30-45 | 45-60 | 60-90 | $90\text{-}\infty$ | All |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CENT [13] | 9.93% | 5.19% | 3.31 | 15.34 | 16.87 | 27.61 | 48.28 | 69.19 | 92.62 | 209.13 | 32.86 |
| CENT+SD [13] | 8.91% | 4.85% | 4.33 | 16.7 | 12.29 | 19.92 | 38.20 | 60.69 | 81.22 | 216.02 | 28.67 |
| CENT+SD / **Inter** | 8.75% | 4.70% | 2.61 | 10.50 | 8.64 | 15.29 | 30.38 | 42.87 | 66.16 | 137.81 | 20.73 |
| Hinge | 7.78% | 5.18% | 1.93 | 8.14 | 5.81 | 10.98 | 31.95 | 50.97 | 73.24 | 185.81 | 21.40 |
| Hinge+SD | 7.74% | 4.85% | 2.32 | 7.32 | 5.32 | 9.38 | 25.21 | 50.43 | 67.32 | 216.39 | 20.51 |
| Hinge+SD / Neg-mining | 7.53% | 5.00% | 3.06 | 6.19 | 5.41 | 10.52 | 26.88 | 51.33 | 70.29 | 210.34 | 20.96 |
| Hinge+SD / Cur. by disp | 7.67% | 4.83% | 2.71 | 8.61 | 5.26 | 10.26 | 14.76 | 48.88 | 65.15 | 220.13 | 20.67 |
| Hinge+SD / Cur. by dist | 7.47% | 4.93% | 2.83 | 8.66 | 5.25 | 10.35 | 23.62 | 45.82 | 63.69 | 197.82 | 19.70 |
| Hinge+SD / Self-Paced | 8.75% | 5.23% | 2.88 | 9.35 | 6.84 | 13.74 | 34.09 | 57.46 | 80.8 | 198.97 | 23.93 |
| Hinge+SD / Anti-Inter | 14.53% | 8.30% | 2.98 | 9.12 | 13.36 | 20.63 | 37.69 | 42.41 | 81.41 | 132.03 | 24.11 |
| Hinge+SD / **Inter** | 6.60% | 4.41% | 1.41 | 5.57 | 3.07 | 6.31 | 15.6 | 28.52 | 43.46 | 127.65 | 12.61 |
| Hinge+SD / **SPCI** | 6.64% | 4.37% | 1.40 | 5.04 | 3.46 | 6.56 | 15.11 | 27.13 | 42.72 | 130.17 | 12.50 |
| Hinge+SD+PS71 | 7.34% | 4.76% | 1.96 | 5.44 | 5.28 | 11.8 | 22.76 | 42.3 | 67.27 | 190.3 | 18.91 |
| Hinge+SD+PS71 / **Inter** | 6.17% | 4.35% | **1.00** | 3.96 | 2.22 | 4.11 | 11.33 | **20.87** | 32.53 | **119.74** | 9.80 |
| Hinge+SD+PS71 / **SPCI** | **6.12%** | **4.27%** | 1.02 | **3.42** | **2.16** | **3.52** | **10.55** | 21.28 | **32.17** | 119.98 | **9.54** |

Table 2. Architecture and learning method comparison by the output error of the PatchMatch and EpicFlow steps in the pipeline and by distractors amount. SD symbols the addition of the standard deviation to the loss function, PS71 is for using a patch size of 71 × 71 pixels. Neg-mining was implemented as described in [33] with a factor of 2. See Section 5 for an explanation of the other learning methods. The error is the percent of pixels in the validation set with euclidean error $>$ 3 pixels. Distractors are calculated as described in Section 4.

pand it by adding a dynamic control on the difficulty level. In order to maintain the category diversity, we simply modify the distance equation for epoch $i$ to:

$$d_i = v(1 - X - R_i) \tag{5}$$

where $R_i$ is define as:

$$R_i = \underbrace{\frac{i}{m}}_{\text{curriculum}} \cdot \underbrace{\max(0, 1 - \frac{l_{i-1}}{l_{init}})}_{\text{self-paced}} \tag{6}$$

and $m$ is the total epoch amount, $l_i$ is the validation loss on epoch $i$ and $l_{init}$ is some initial loss to compare. We defined $l_{init}$ as the loss on epoch number 5. Until that epoch, self-pacing is not applied.

The curriculum addition enhances the global difficulty of false samples in each iteration by shorting the taken distance and, therefore, integrates an instructor-driven approach assuming the student will handle more difficult tasks with time. To add a student-driven portion, we use the self-paced component which allows a feedback from the model to influence the difficulty of the next iteration. Integrating all of this together, we get a learning method that learns all strategies simultaneously and in which the difficulty is increased over iterations and with a success feedback.

## 6. Experiments

We perform two families of experiments. First, MNIST recognition experiments are presented as a testbed for the learning schemes. Then, the main set of experiments is performed on the specific problem of optical flow.

### 6.1. MNIST

In order to validate our learning methods on a task different from optical flow, we used the MNIST handwritten digit database [25]. This data set consists of images showing a digit from 0 to 9 with their true label. We divided the data into two different classes – class L contains digits $0..4$ and class H contains $5..9$ . To enable difficulty differentiation between samples, random noise was added to the top half of the images of $H$ and to the bottom part of the $L$ images. Furthermore, images from class $H$ were rotated by a random angle of $[0, 45]$ degrees with correlation to the noise amount, such that, samples that are more noisy are also rotated in larger angles.

While referring noisier samples as harder, we trained a model using several methods. As curriculum learning, harder samples were added to the training pool in each epoch. In the self-paced model, the hardness of the samples to learn from was derived from the loss. Interleaving was implemented by using all of the noise range level in each epoch with a fewer noised samples for the harder $H$ class against more for $L$ class. An integration of interleaving with Curriculum and Self-Paced methods was also used by increasing the the amount of the noised $H$ samples in each epoch. As can be seen in Tab. 3, interleaving produced the greatest improvement and SPCI attained the best results.

### 6.2. Optical flow

To evaluate our work, we use the three most competitive optical flow benchmarks - KITTI2012 [15], KITTI2015 [29] and MPI-Sintel [8]. We use their data to

| Method | $L$ | $H$ |
|---|---|---|
| Random order | 97.98% | 82.24% |
| Curriculum | 98.10% | 87.89% |
| Self-Paced | 98.26% | 88.33% |
| **Interleaving** | 98.26% | 95.00% |
| **Interleaving**+Curriculum | 98.30% | 95.62% |
| **Interleaving**+SP | 98.14% | 95.31% |
| **SPCI** | **98.38%** | **96.33%** |

Table 3. The improvement of results on the MNIST experiment using interleaving methods. Column $L$ shows the results on digits $[0, 4]$ with random noise on the image bottom, and column $H$ shows the results on digits $[5, 9]$ rotated randomly by $0$ to $45$ degrees with random noise at the top of the image .

conduct a series of experiments to measure the effect of each of our contributions and to submit our best results to compare with other methods.

By training the different models on a subset of 80% from the KITTI2012 dataset for 500 epochs and testing the results on the remaining 20% image pairs, we show a comparison of the models summarized in Tab. 2. Note that lower PatchMatch (PM) error is not always correlated with lower EpicFlow (EF) error because of the bidirectional consistency check that excludes some inconsistent results to generate a sparse flow as an input for EF.

Observing Tab. 2, one can notice that the use of the Hinge loss instead of CENT [13], improved the PM results and has no such effect on the final EF output. However, combining with the batch standard deviation term (SD) and our interleaving learning (Inter) leads to an advantage of the Hinge loss. Our interleaving learning method outperforms both Curriculum learning and Self-Paced learning. The SPCI technique contributes an additional improvement.

Integrating all of our architecture modifications with SPCI produces the lowest error percent on the validation set with a major improvement on the initial baseline. Moreover, the amount of nearby distractors with descriptors that are more similar to the original patch than the true match is reduced to one third of the baseline.

As a sanity-check experiment we evaluate an *Anti-Interleaving* method. In this method, negative matches from different ranges were also used. However, the ratio was inverted – true matches of small displacements were matched with false samples with large distances and vice versa. The high error of this model, as can be seen in Tab. 2, implies that the use of different ranges for false matches was not the main benefit of the interleaving method and it is the correlation with the displacement values that is the crucial factor.

We also experimented with hard-negative mining [33] and concluded that its benefits are limited because, unlike the interleaving method, it might neglect some displacement ranges during the train.

| Method | 5 - 10 | 10 - 40 | 40 - $\infty$ |
|---|---|---|---|
| Baseline | 95.01% | 97.61% | 97.83% |
| Cur. by displacement* | 96.82% | 98.56% | 101.04% |
| Cur. by distance* | 98.40% | 98.32% | 100.29% |
| Self-Paced* | 93.66% | 93.67% | 99.78% |
| Anti-Interleaving | 105.29% | 116.34% | 103.26% |
| **Interleaving** | 97.32% | 94.67% | 93.71% |
| **Interleaving**+Cur.** | 96.40% | 95.39% | 95.24% |
| **Interleaving**+SP** | 95.82% | 95.38% | 93.61% |
| **SPCI** | 96.02% | 92.66% | 90.11% |

Table 4. Learning method comparison by descriptor sensitivity to location movement for different displacement ranges, measured by dividing the average distance of the descriptors of 5 pixels neighbor patches associated with a certain displacement range with the average obtained at for displacements smaller than 5 pixels. Methods marked with * were implemented as described in the beginning of Section 5 and the ones marked with ** were trained like SPCI, but applying only one multiplier in Eq. 6. Using only gradual methods seems not to have any tendency relating to displacement value. In contrast, the interleaving models have learned to progressively decrease sensitivity for larger values.

### 6.2.1 Sensitivity to appearance change

Part of what the networks learn is to behave differently to patches with different *expected* displacements. Those patches that are similar to patches that are associated with small displacements are treated differently than those which were associated, in the training set, with large displacements. To illustrate this, and compare the various learning methods, we explore the model behavior on nearby patches from the *same image* for varied displacement ranges. First, we measure the average distance $\bar{d}_{0-5}$ of a patch descriptor from that of a patch that is 5 pixels away for pixels which undergo a displacement of up to 5 pixels. Note that for a $51 \times 51$ patch, only 18% of the pixels were completely replaced in such a small displacement. Then, we repeat this to patches from various displacement ranges, taking again the average distance from a patch of 5 pixels away. To normalize, we divide this average distance by the first average $\frac{\bar{d}_{L-H}}{\bar{d}_{0-5}}$, for $(L, H) \in \{(5, 10), (10, 40), (40, \inf)\}$.

The results in Tab. 4 show that while the PatchBatch original model reacts almost similarly for all displacement ranges, interleaving trained models have learned to be less sensitive to appearance changes for larger displacements. Moreover, using only gradual learning, leads to high sensitivity across all ranges. This can be the result of the carry-on from the early learning stages on small displacements where appearance sensitivity is more valuable.

### 6.2.2 Benchmarks results

We train our model on three datasets and submit the results of each benchmark on the respectively trained model. Our

| Method | Out-Noc |
|---|---|
| **Imp. PatchBatch+SPCI** | **4.65%** |
| CNN-HPM [3] | 4.89% |
| **Imp. PatchBatch** | 4.92% |
| PatchBatch+PS71 [13] | 5.29% |
| PatchBatch [13] | 5.44% |
| PH-Flow [40] | 5.76% |
| FlowFields [2] | 5.77% |
| CPM-Flow [18] | 5.79% |

Table 5. Top 8 published KITTI2012 Pure Optical Flow methods as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in Section 3. Out-Noc is the percentage of pixels with euclidean error $>$ 3 pixels out of the non-occluded pixels.

| Method | Fl-bg | Fl-fg | Fl-all |
|---|---|---|---|
| **Imp. PatchBatch+SPCI** | **17.25%** | **24.52%** | **18.46%** |
| CNN-HPM [3] | 18.90% | 24.96% | 19.44% |
| PatchBatch [13] | 19.98% | 30.24% | 21.69% |
| DiscreteFlow [30] | 21.53% | 26.68% | 22.38% |
| CPM-Flow [18] | 22.32% | 27.79% | 23.23% |
| FullFlow [9] | 23.09% | 30.11% | 24.26% |
| EpicFlow [32] | 25.81% | 33.56% | 27.10% |
| DeepFlow [39] | 27.96% | 35.28% | 29.18% |

Table 6. Top 8 published KITTI2015 Pure Optical Flow methods as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in Section 3. Fl-all is the percentage of outliers (pixels with euclidean error $>$ 3 pixels). Fl-bg, Fl-fg are the percentage of outliers only over background and foreground regions respectively.

results are directly comparable with the PatchBatch model, since we use the same procedure as theirs – Training the CNN for 4000 epochs on 80% of the training set and choosing the best configuration by selecting the one with the lowest validation error on samples from the remaining 20% of the data.

The results can be seen in Tab. 5, 6, 7. We succeed in improving results in all three benchmarks and achieve state of the art results for KITTI2012 [15] and KITTI2015 [29].

We evaluate our method only against methods not using additional information for the flow estimation, including those methods which used semantic segmentation.

On KITTI2015, as can be seen on Tab. 6, we reduced the error of both foreground and background areas, obtaining the lowest error for both cases. The increased accuracy for both regions is correlated with our previous experiments and corroborate our claim of extracting better descriptors for all scenarios.

In contrast to the error percent measurement of the KITTI benchmarks, MPI-Sintel uses an end-point-error (EPE) one. Compared to the original PatchBatch model, (Tab. 7) we succeed in preserving a low EPE for small displacements while significantly reducing it for large ones. Our model does not achieve the best results when using the EPE measurement. However, when considering the percentage of large error displacements, as calculated from the error images, our SPCI model is second best and our interleaving model is third.

| Method | EPE | Fl | s0-10 | s40+ |
|---|---|---|---|---|
| FlowFields+ [2] | **5.71** | 8.14% | 1.31 | **34.17** |
| DeepDiscreteFlow [16] | 5.73 | **7.30%** | 0.96 | 35.82 |
| SPM-BPv2 [26] | 5.81 | 9.17% | 1.05 | 35.12 |
| FullFlow [9] | 5.90 | 9.55% | 1.14 | 35.59 |
| CPM-Flow [18] | 5.96 | 8.31% | 1.15 | 35.14 |
| GlobalPatchCollider [38] | 6.04 | 10.21% | 1.10 | 36.45 |
| DiscreteFlow [30] | 6.08 | 9.52% | 1.07 | 36.34 |
| **Imp. PatchBatch+Inter** | 6.22 | 8.11% | 0.91 | 39.91 |
| **Imp. PatchBatch+SPCI** | 6.24 | 7.89% | 0.88 | 40.07 |
| EpicFlow [32] | 6.28 | 11.26% | 1.13 | 38.02 |
| FGI [27] | 6.61 | 12.34% | 1.15 | 39.98 |
| TF+OFM [20] | 6.73 | 11.35% | 1.51 | 39.76 |
| Deep+R [12] | 6.77 | 13.71% | 1.16 | 41.69 |
| PatchBatch [13] | 6.78 | 8.66% | **0.72** | 45.86 |

Table 7. Comparison of our models with the top methods for the MPI-Sintel benchmark as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in Section 3. The EPE (end-point-error) is averaged over all the pixels and the two right columns contain only the EPE of pixels within the displacement range mentioned in the title. The Fl column presents an evaluation of the the outlier percentage, which, although not provided by this benchmark, was calculated from the error figures presented for each scene that have higher pixel values for larger errors. Fl is the percentage of pixels with a value larger than 120.

Our trained models are available on the PatchBatch GitHub repository.

## 7. Conclusions

Common sense dictates that most of the perceptual tasks are heterogeneous and require multiple strategies. The literature methods address training in accordance with the difficulty of specific samples. In our work, we show, for the first time, how to address both multiple sub-tasks and varying difficulty. The two are not independent – some sub-tasks are harder than others, and our interleaving methods address this challenge.

Using the proposed novel methods, we are able to improve a recently proposed optical flow model and obtain state of the art results on the two most competitive real-world benchmarks.

## Acknowledgments

This research is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

## References

[1] G. Antipov, M. Baccouche, S.-A. Berrani, and J.-L. Dugelay. Apparent age estimation from face images combining general and children-specialized deep learning models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[2] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 4015–4023, 2015.

[3] C. Bailer, K. Varanasi, and D. Stricker. Cnn based patch matching for optical flow with thresholded hinge loss. *arXiv preprint arXiv:1607.08064*, 2016.

[4] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision (ECCV)*, pages 29–43. Springer, 2010.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision (ECCV)*, pages 404–417. Springer, 2006.

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[7] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[8] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conference on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[9] Q. Chen and V. Koltun. Full flow: Optical flow estimation by global optimization over regular grids. *arXiv preprint arXiv:1604.03513*, 2016.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[11] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766. IEEE, 2015.

[12] B. Drayer and T. Brox. Combinatorial regularization of descriptor matching for optical flow estimation. In *British Machine Vision Conference (BMVC)*, volume 8, 2015.

[13] D. Gadot and L. Wolf. Patchbatch: A batch augmented loss for optical flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[14] Z. Ge, A. Bewley, C. McCool, P. Corke, B. Upcroft, and C. Sanderson. Fine-grained classification via mixture of deep convolutional neural networks. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 1–6. IEEE, 2016.

[15] 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.

[16] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[17] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742. IEEE, 2006.

[18] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patchmatch for large displacement optical flow.

[19] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann. Self-paced curriculum learning. 2015.

[20] R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (CVPR)*, pages 364–377. Springer, 2015.

[21] N. Kornell and R. A. Bjork. Learning concepts and categories is spacing the enemy of induction? *Psychological science*, 19(6):585–592, 2008.

[22] J. Krause, T. Gebru, J. Deng, L.-J. Li, and F.-F. Li. Learning features and parts for fine-grained recognition. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, page 8. Citeseer, 2014.

[23] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5546–5555, 2015.

[24] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. Curran Associates, Inc., 2010.

[25] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[26] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu. Spm-bp: Sped-up patchmatch belief propagation for continuous mrfs. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4006–4014, 2015.

[27] Y. Li, D. Min, M. N. Do, and J. Lu. Fast guided global interpolation for depth and motion. In *European Conference on Computer Vision (ECCV)*, pages 717–733. Springer, 2016.

[28] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE, 1999.

[29] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition*, pages 16–28. Springer, 2015.

[31] R.-D. Reiss and M. Thomas. *Statistical Analysis of Extreme Values: with Applications to Insurance, Finance, Hydrology*

*and Other Fields*, pages 31–32. Birkhuser Basel, 3rd ed. edition, 2007.

[32] 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 (CVPR)*, pages 1164–1172, 2015.

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

[34] R. Timofte and L. Van Gool. Sparse flow: Sparse matching for small to large displacement optical flow. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 1100–1106. IEEE, 2015.

[35] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010.

[36] E. Walach and L. Wolf. *Learning to Count with CNN Boosting*, pages 660–676. Springer International Publishing, Cham, 2016.

[37] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2443–2451, 2015.

[38] S. Wang, S. Ryan Fanello, C. Rhemann, S. Izadi, and P. Kohli. The global patch collider. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[39] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013.

[40] J. Yang and H. Li. Dense, accurate optical flow estimation with piecewise parametric model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1019–1027, 2015.

[41] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, 2015.