# Weekly Report II
## for Laboratory Research

University of Houston

May 26, 2018
and
Jun. 1, 2018

# Outline

# Outline

# Personal site is ready

- My personal site is ready during the two weeks.
- The DGX Work Station is ready. We have equipped it with Matlab, Tensorflow and Docker. To be specific, I have written to tutorials for it:
  - How to access to the DGX server: *Basic Linux Skills for Remote Controlling*. Check it!
  - How to manage the installed packages: *Advanced Linux Skills for Using NVIDIA Docker*. Check it!
- A detailed version of this note could be seen here: Check it!

Site

Set-invariant network

Theory

Results

Style Transfer

Semi-supervised learning

Reference

# Outline

Site

Set-invariant network

Theory

Results

Style Transfer

Semi-supervised learning

Reference

# Set-invariant network
## Deep Sets [1]

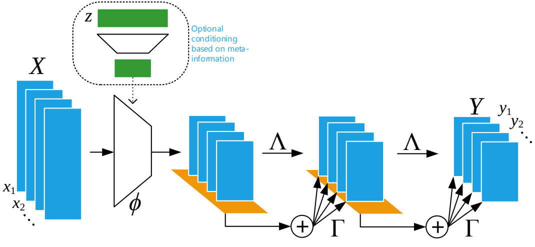



Figure 1: Deep Sets architecture.

- Stacked structure by repeating the set-invariant layer.
- Each layer accepts a input set and give the corresponding output set.

# Set-invariant network

## Deep Sets

### The net layer specification

Use a diagonal kernel $\mathbf{\Gamma}$ and a bias vector $\boldsymbol{\beta}$ to define a layer.

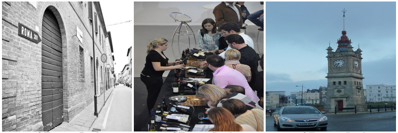
$$F(\mathbf{x},\ \mathbf{\Gamma},\ \boldsymbol{\beta}) = \sigma\left(\boldsymbol{\beta} + (\mathbf{x} - \mathbf{1} \cdot \mathrm{maxpool}(\mathbf{x}))\,\mathbf{\Gamma}\right). \tag{1}$$

### The probability view

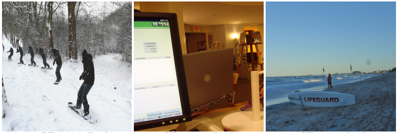
This layer could be viewed by deducing the *de Finetti*'s Theorem. We use $\mathbb{X}$ to represent the input set, $\theta$ is the latent feature and $\alpha,\ M_0$ are the hyper-parameters of the prior.

$$\begin{aligned} p(\mathbb{X}|\alpha,\ M_0) &= \int \left[\prod_{m=1}^{M} p(x_m|\theta)\right] p(\theta|\alpha,\ M_0)\mathrm{d}\theta \\ &= e^{h(\alpha+\phi(\mathbb{X}),\ M+M_0) - h(\alpha,\ M_0)}. \end{aligned} \tag{2}$$

# Set-invariant network
## Results




| GT | Pred | GT | Pred | GT | Pred |
|---|---|---|---|---|---|
| building | building | standing | person | traffic | clock |
| sign | street | surround | group | city | tower |
| brick | city | woman | man | building | sky |
| picture | brick | crowd | table | tall | building |
| empty | sidewalk | wine | sit | large | tall |
| white | side | person | room | tower | large |
| black | pole | group | woman | European | cloudy |
| street | white | table | couple | front | front |
| image | stone | bottle | gather | clock | city |




| GT | Pred | GT | Pred | GT | Pred |
|---|---|---|---|---|---|
| photograph | ski | laptop | refrigerator | beach | jet |
| snowboarder | snow | person | fridge | shoreline | airplane |
| snow | slope | screen | room | stand | propeller |
| glide | person | room | magnet | walk | ocean |
| hill | snowy | desk | cabinet | sand | plane |
| show | hill | living | kitchen | lifeguard | water |
| person | man | counter | shelf | white | body |
| slope | skiing | computer | wall | person | person |
| young | skier | monitor | counter | surfboard | sky |

Figure 2: Result of the set-invariant classification.

# Outline

# Style Transfer

## Image Style Transfer Using Convolutional Neural Networks [2]

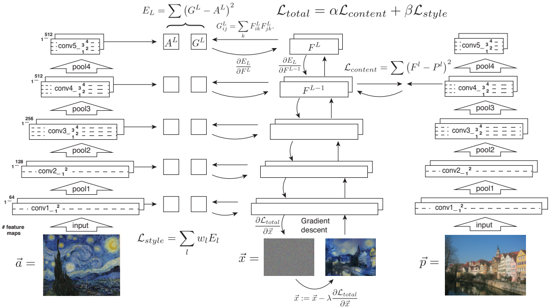



Figure 3: Architecture of optimization method.

- Use a pre-trained and fixed network to extract features.
- Use Gramian matrix (pre-defined method) to extract the texture features.
- Optimize the input image to reduce the conjugated loss function.

# Style Transfer

## Image Style Transfer Using Convolutional Neural Networks

### The conjugated loss

The conjugated loss is composed of content loss and style loss.

$$\mathbf{x} = \arg\min_{\boldsymbol{\theta}} \alpha \mathscr{L}_c(\boldsymbol{\theta}, \mathbf{x}_c) + \beta \mathscr{L}_s(\boldsymbol{\theta}, \mathbf{x}_s). \tag{3}$$

### The content loss

The content loss is from the output of one layer (we use $\mathscr{F}^{(l)}$ to represent the output features of the $l^{\text{th}}$ layer).

$$\mathscr{L}_c = \|\mathscr{F}^{(L)}(\boldsymbol{\theta}) - \mathscr{F}^{(L)}(\mathbf{x}_c)\|_2^2. \tag{4}$$

# Style Transfer

## Image Style Transfer Using Convolutional Neural Networks

### The conjugated loss

The conjugated loss is composed of content loss and style loss.

$$\mathbf{x} = \arg\min_{\boldsymbol{\theta}} \alpha \mathscr{L}_c(\boldsymbol{\theta},\ \mathbf{x}_c) + \beta \mathscr{L}_s(\boldsymbol{\theta},\ \mathbf{x}_s). \tag{3}$$

### The content loss

The style loss is from feature maps of all layers.

$$\begin{aligned} \mathscr{L}_s &= \sum_l w_l \|\mathscr{G}^{(l)}(\boldsymbol{\theta}) - \mathscr{G}^{(l)}(\mathbf{x}_s)\|_2^2, \\ \mathscr{G}^{(l)}(\mathbf{x})_{ij} &= \frac{1}{K} \sum_k \mathscr{F}^{(l)}(\mathbf{x})_{ik} \mathscr{F}^{(l)}(\mathbf{x})_{jk}. \end{aligned} \tag{4}$$

Site

Set-invariant network

Style Transfer

Theory

Results

Semi-supervised learning

Reference

# Style Transfer

## Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization [3]

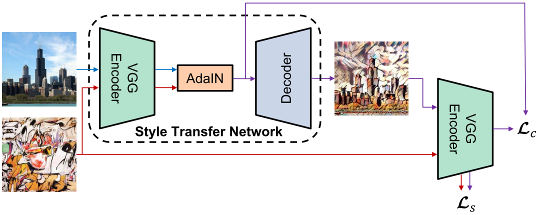



Figure 4: Architecture of normalization method.

- Use a pre-trained auto-encoder network. Fix the encoder while train the decoder.
- Replace the mean and std. value of the encoded content features with that of the style features.
- The mean and std value is calculated by instance normalization.

# Style Transfer

## Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

### The net layer specification

The loss function is also composed of content loss and style loss. We use $\mathbf{\Theta}_D$ to represent the parameters of the decoder.

$$\arg\min_{\mathbf{\Theta}_D} \mathscr{L}_c(\mathbf{x}_c,\ \mathbf{x}_s,\ \mathbf{\Theta}_D) + \lambda \mathscr{L}_s(\mathbf{x}_c,\ \mathbf{x}_s,\ \mathbf{\Theta}_D). \tag{5}$$

### The content loss

$$\mathscr{L}_c = \|E(D(\hat{\mathbf{y}})) - \hat{\mathbf{y}}\|_2^2. \tag{6}$$

$\hat{\mathbf{y}}$ is the encoded features whose mean and std. get replaced by that of the encoded style features.

$$\hat{\mathbf{y}} = \sigma(E(\mathbf{x}_s))\left(\frac{E(\mathbf{x}_c) - \mu(E(\mathbf{x}_c))}{\sigma(E(\mathbf{x}_c))}\right) + \mu(E(\mathbf{x}_s)). \tag{7}$$

# Style Transfer

## Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

### The net layer specification

The loss function is also composed of content loss and style loss. We use $\Theta_D$ to represent the parameters of the decoder.

$$\arg\min_{\Theta_D} \mathscr{L}_c(\mathbf{x}_c,\ \mathbf{x}_s,\ \Theta_D) + \lambda \mathscr{L}_s(\mathbf{x}_c,\ \mathbf{x}_s,\ \Theta_D). \tag{5}$$

### The style loss

$$\begin{aligned} \mathscr{L}_s = & \sum_l \|\mu(E^{(l)}(D(\hat{\mathbf{y}}))) - \mu(E^{(l)}(\mathbf{x}_s)\|_2^2 \\ & + \sum_l \|\sigma(E^{(l)}(D(\hat{\mathbf{y}}))) - \sigma(E^{(l)}(\mathbf{x}_s))\|_2^2. \end{aligned} \tag{6}$$

# Style Transfer

## Universal Style Transfer via Feature Transforms [4]



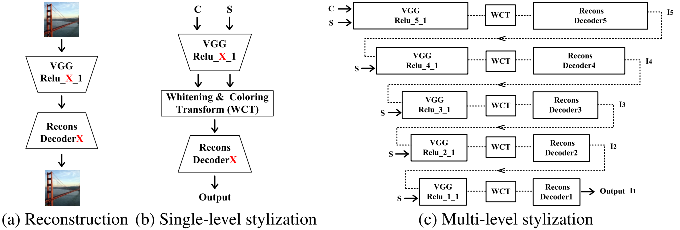



Figure 5: WCT architecture.

- Use pre-trained and fixed auto-encoder network to extract the feature.
- Perform the Whitening and Coloring Transformation (WCT) on features to get style converted.

# Style Transfer

## Universal Style Transfer via Feature Transforms

### Features

Use mean shifted features and decompose its covariance matrix.

$$\mathbf{y} = E^{(l)}(\mathbf{x}) - \mu(E^{(l)}(\mathbf{x})), \qquad \mathbf{y}\mathbf{y}^T = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T. \tag{7}$$

### The whitening transformation

Remove the style feature by whitening.

$$\hat{\mathbf{y}}_c = \mathbf{Q}_c\boldsymbol{\Lambda}_c^{-\frac{1}{2}}\mathbf{Q}_c^T\mathbf{y}_c. \tag{8}$$

### The coloring transformation

Add the style feature by coloring.

$$\mathbf{y} = \mathbf{Q}_s\boldsymbol{\Lambda}_s^{\frac{1}{2}}\mathbf{Q}_s^T\hat{\mathbf{y}}_c. \tag{9}$$

# Style Transfer
## Results

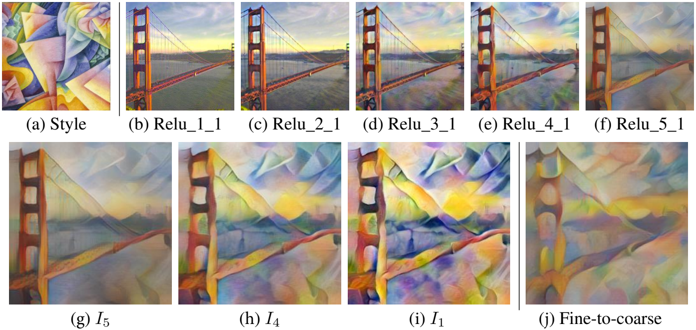



Figure 6: Using different layers' features to perform WCT.

# Style Transfer
## Results



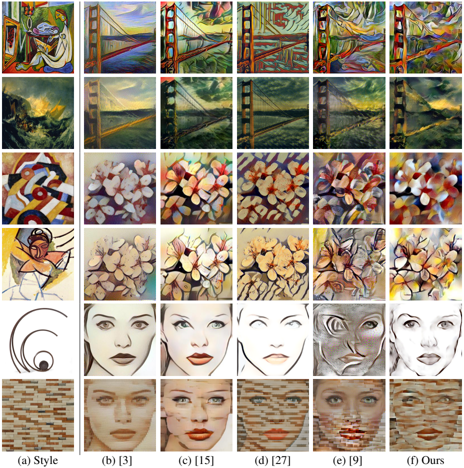
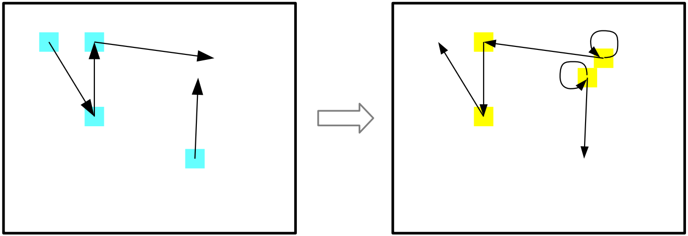



Figure 6: Compare the performance of style transferring methods.

# Outline

# Semi-supervised learning
## Spatial Transformer Networks [5]





$\mathbf{I}(t)$ $\qquad$ $\mathbf{I}_{\text{warp}}(t+1)$

Figure 7: Differentiable image warp method.

- Propose a differentiable interpolation method for image warping.
- Extend the affine transformation method.

# Semi-supervised learning

## Spatial Transformer Networks

### Affine transformation

Use mean shifted features and decompose its covariance matrix.

$$\begin{pmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \end{pmatrix} = \mathbf{W}_{ij} \begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} = \begin{bmatrix} u_{ij} & 0 \\ 0 & v_{ij} \end{bmatrix} \begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} = \begin{pmatrix} x_{ij} + u_{ij} \\ y_{ij} + v_{ij} \end{pmatrix} \tag{10}$$

### Differentiable Warp

$$\mathbf{I}_{\mathrm{warp}}(x_{ij},\ y_{ij},\ t) = \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{I}(h,\ w,\ t+1)\mathrm{M}(1 - |\hat{x}_{ij} - w|)\mathrm{M}(1 - |\hat{y}_{ij} - h|), \tag{11}$$

where $\mathrm{M}(\cdot) = \max(0,\ \cdot)$.

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness [6]

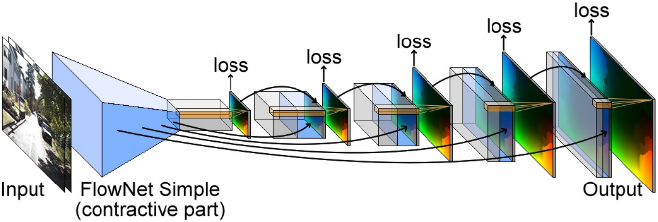



Figure 8: FlowNet architecture.

- The baseline network is auto-encoder.
- Each layer of the decoder is optimized to the prediction flow in different scale.
- The flow is optimized for both photometric target and smoothness.

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness

### The whole loss

The loss function is composed of photometric loss and smoothness loss. We use $D_l^{Cn}$ represent the $n^{\text{th}}$ channel of the $l^{\text{th}}$ layer of the up-sampling features (decoder output).

$$\mathscr{L}_{\text{total}} = \sum_{l} \ell_p(\mathbf{u},\ \mathbf{v},\ \mathbf{I}(t),\ \mathbf{I}(t+1)) + \lambda \ell_s(\mathbf{u},\ \mathbf{v})\Big|_{\mathbf{u}=D_l^{C1},\ \mathbf{v}=D_l^{C2}}. \quad (12)$$

### The photometric loss

The photometric loss is used to control the warp error between real frame and predicted (interpolated) frame.

$$\ell_p = \sum_{xy} \rho_D(\mathbf{I}(t) - \mathbf{I}_{\text{warp}}(t)). \quad (13)$$

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness

### The whole loss

The loss function is composed of photometric loss and smoothness loss. We use $D_l^{Cn}$ represent the $n^{\text{th}}$ channel of the $l^{\text{th}}$ layer of the up-sampling features (decoder output).

$$\mathscr{L}_{\text{total}} = \sum_l \ell_p(\mathbf{u},\ \mathbf{v},\ \mathbf{I}(t),\ \mathbf{I}(t+1)) + \lambda \ell_s(\mathbf{u},\ \mathbf{v})\Big|_{\mathbf{u}=D_l^{C1},\ \mathbf{v}=D_l^{C2}} . \quad (12)$$

### The smoothness loss

The smooth loss is used to reduce the roughness of the flow prediction.

$$\ell_s = \sum_{xy} \left[ \rho_S\left(\frac{\partial \mathbf{u}}{\partial x}\right) + \rho_S\left(\frac{\partial \mathbf{u}}{\partial y}\right) + \rho_S\left(\frac{\partial \mathbf{v}}{\partial x}\right) + \rho_S\left(\frac{\partial \mathbf{v}}{\partial y}\right) \right] . \quad (13)$$

# Semi-supervised learning

## Semi-Supervised Learning for Optical Flow with Generative Adversarial Networks [7]

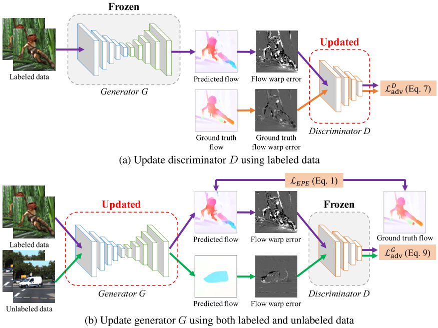



Figure 9: GAN based learning architecture.

- Only use labeled data to train the discriminator.
- Use both labeled and unlabeled data to train the generator.
- Use the warp loss from the previous to realize the unsupervised learning part.

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness

### The whole loss

The loss function is composed of a supervised learning loss ($\mathscr{L}_s$) and an adversarial loss ($\mathscr{L}_a$). We use $G$ and $D$ to denote the generator and discriminator respectively.

$$\min_G \max_D \mathscr{L}_s(G) + \lambda \mathscr{L}_a(G,\ D). \tag{14}$$

### The discriminator loss

When training the discriminator, we reduce the warp loss which comes from $\hat{\mathbf{y}} = \mathbf{I}_t - \mathscr{W}(\mathbf{I}_{t+1},\ \mathbf{g})$ and $\mathbf{y} = \mathbf{I}_t - \mathscr{W}(\mathbf{I}_{t+1},\ \mathbf{g}_0)$, where we use $\mathscr{W}$ to represent the mentioned differentiable warping. We use predicted flow to get $\hat{\mathbf{y}}$ and ground truth to get $\mathbf{y}$.

$$\mathscr{L}_a^D(\mathbf{I}_t,\ \mathbf{I}_{t+1},\ \mathbf{g}_0) = -\log D(\hat{\mathbf{y}}) - \log(1 - D(\mathbf{y})). \tag{15}$$

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness

### The whole loss

The loss function is composed of a supervised learning loss ($\mathscr{L}_s$) and an adversarial loss ($\mathscr{L}_a$). We use $G$ and $D$ to denote the generator and discriminator respectively.

$$\min_{G} \max_{D} \mathscr{L}_s(G) + \lambda \mathscr{L}_a(G,\ D). \tag{14}$$

### The generator loss (supervised)

When we use labeled data to optimize the generator, the supervised learning loss contains a loss from ground truth and an adversarial loss.

$$\mathscr{L}_{\text{sup}}^{G} = \|G(\mathbf{I}_t,\ \mathbf{I}_{t+1}) - \mathbf{g}_0\|_F - \lambda \log D(\hat{\mathbf{y}}). \tag{15}$$

# Semi-supervised learning

## Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness

### The whole loss

The loss function is composed of a supervised learning loss ($\mathscr{L}_s$) and an adversarial loss ($\mathscr{L}_a$). We use $G$ and $D$ to denote the generator and discriminator respectively.

$$\min_G \max_D \mathscr{L}_s(G) + \lambda \mathscr{L}_a(G,\ D). \tag{14}$$

### The generator loss (unsupervised)

When we use unlabeled data to optimize the generator, The unsupervised learning loss only contains an adversarial loss.

$$\mathscr{L}^G_{\text{sup}} = -\lambda \log D(\hat{\mathbf{y}}). \tag{15}$$

# Semi-supervised learning

## Results

| Method | Training Datasets | Sintel-Clean EPE | Sintel-Final EPE | KITTI 2012 EPE | KITTI 2015 EPE | KITTI 2015 Fl | FlyingChairs EPE |
|---|---|---|---|---|---|---|---|
| Supervised | Chairs | 3.51 | 4.70 | 7.69 | 17.19 | 40.82% | 2.15 |
| Unsupervised | KITTI | 8.01 | 8.97 | 16.54 | 25.53 | 54.40% | 6.66 |
| Baseline semi-supervised | Chairs + KITTI | 3.69 | 4.86 | 10.38 | 18.07 | 39.33% | 2.11 |
| Proposed semi-supervised | Chairs + KITTI | **3.30** | **4.68** | **7.16** | **16.02** | **38.77%** | **1.95** |

Figure 10: Numerical comparison among different methods.

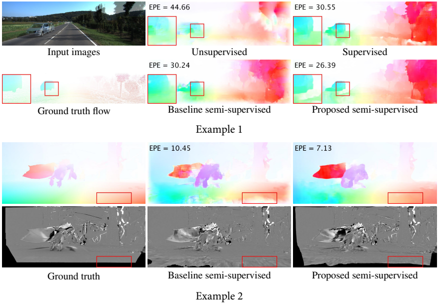



Figure 11: Illustrated comparison among different methods.

# Outline

Site

Set-invariant network

Style Transfer

Semi-supervised learning

**Reference**

# Reference I


- M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," *CoRR*, vol. abs/1703.06114, 2017. [Online]. Available: http://arxiv.org/abs/1703.06114
- L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2414–2423.
- X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization," *ArXiv e-prints*, Mar. 2017.
- Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal Style Transfer via Feature Transforms," *ArXiv e-prints*, May 2017.
- M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," *ArXiv e-prints*, Jun. 2015.

Site

Set-invariant network

Style Transfer

Semi-supervised learning

**Reference**

# Reference II


J. J. Yu, A. W. Harley, and K. G. Derpanis, “Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness,” *ArXiv e-prints*, Aug. 2016.

W.-S. Lai, J.-B. Huang, and M.-H. Yang, “Semi-supervised learning for optical flow with generative adversarial networks,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 354–364. [Online]. Available: http://papers.nips.cc/paper/6639-semi-supervised-learning-for-optical-flow-with-generative-advers pdf