Top-down Flow Transformer Networks

Zhiwei Jia [1] Haoshen Hong [1] Siyang Wang [1] Zhuowen Tu [1]

Abstract

We study the deformation fields of feature maps across convolutional network layers under explicit top-down spatial transformations. We propose top-down flow transformer (TFT) by focusing on three transformations: translation, rotation, and scaling. Flow transformation generators, under controllable parameters, are learned that are able to account for the hidden layer deformations while maintaining the overall consistency across layers. The learned generators capture the underlying feature transformation processes that are independent of the particular training images, demonstrated by a comprehensive study on various datasets including MNIST, shapes, and natural images. Our proposed TFT framework brings insights to and helps the understanding of, the important problem of studying the CNN internal feature representation and transformation under the top-down processes. TFT demonstrates its significant advantage over existing data-driven approaches in building data-independent transformations and it can also be used in applications such as data augmentation and transfer learning.

1. Introduction

Recently, deep neural networks (Krizhevsky et al., 2012; He et al., 2016) have led to tremendous performance improvement on large-scale image classification (Russakovsky et al., 2015b) and other computer vision applications (Girshick et al., 2014; Goodfellow et al., 2014; Long et al., 2015; Xie & Tu, 2015; Dosovitskiy et al., 2015b). While Convolutional Neural Networks (CNNs) have shown great promise in solving many challenging vision problems, there remain fundamental questions about the transparency of representations in current CNN architectures. While the explicit role of the top-down process is a critical issue in perception and cognition, it has received less attention within the current CNN literature.

[1]University of California, San Diego. Correspondence to: Zhiwei Jia <zjia@ucsd.edu>, Haoshen Hong <h7hong@ucsd.edu>, Siyang Wang <siw030@ucsd.edu>, Zhuowen Tu <ztu@ucsd.edu>.

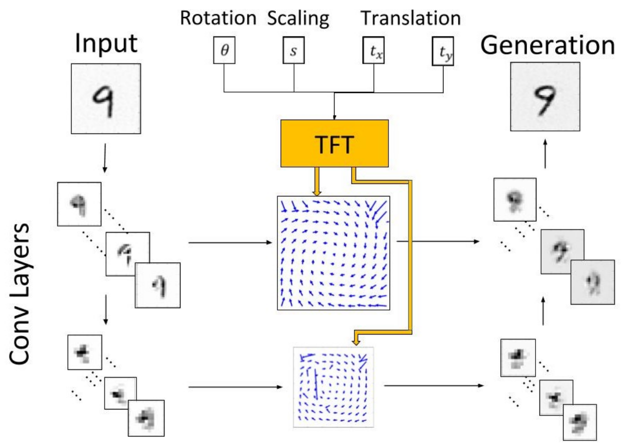


Figure 1. A schematic illustration of our top-down flow transformer framework (TFT).

Currently, both training and testing of CNNs is performed in a data-driven manner by passing convolved features from lower layers to the top layers. However, visual perception systems are shown to engage both bottom-up and top-down processes (Ridley Stroop, 1992; Hill & Johnston, 2007). A top-down process would allow explicit generation and inference of transformations and (high level) configuration changes of images that is otherwise not convenient in a bottom-up process. For example, suppose we wish to train a CNN classifier to detect the translation of an object in an image. A data-driven way to train this CNN would require generating thousands of samples by moving the object around in the image. However, a top-down model, if available, can directly detect translation using two parameters of the translation. Computational models realizing the bottom-up and top-down visual inference have been previously proposed (Marr, 2010; Kersten et al., 2004; Tu et al., 2003). They, however, are not readily combined into end-to-end deep learning frameworks. Recurrent neural networks (RNN) (Elman, 1991; Hochreiter & Schmidhuber, 1997) have feedbacks recursively propagated between the output and input layers but miss explicit top-down generation.

Motivated by the recent development in CNNs (Krizhevsky et al., 2012) that learn effective hierarchical representations, the general pattern theory (Grenander, 1993; Yuille et al., 1992; Zhu et al., 2007) that provides rigorous mathematical formulation for top-down generations, as well as findings from cognitive perception (Gregory, 1980; Dodwell, 1983; Gibson, 2002), we seek to build a top-down generator, **under controllable parameters**, that operates directly on the feature maps of the internal CNN layers to model and account for spatial transformations.

In this paper, we pay particular attention to CNN features under top-down spatial transformations. There often exists clear flow fields computed between the convolutional layers of the original image and the convolutional layers of the transformed images (after rotation, scaling, and translation) (Gallagher et al., 2015). There is a clear pattern of consistent but nontrivial feature map deformations throughout the convolutional layers which is the key topic to be studied and leveraged here. Our goal is to discover and model operations in CNNs that lead to non-linear activity of the resulting flow fields. Given a source image and a transformed image under translation, rotation, and scaling, the internal CNN feature maps across multiple-layers can be directly computed; flow transformers modeled using an aggregated convolution strategy are themselves learned to perform mappings that transfer the feature maps of source image to the target image across all the intermediate CNN layers.

The training process is supervised since we generate transformed images using different parameters for translation, rotation, and scaling. Given the fact that no supervision is needed to have the explicit correspondences at the feature maps level and that transforming the source images can be readily accomplished by using the explicit transformation parameters, obtaining the training data can be done automatically. The learned top-down flow transformer (TFT) however demonstrates great generalization capable of transforming images that are not seen in the training set — a benefit of having a top-down generator that is not tied to specific images. TFT is therefore distinct from existing work (Dosovitskiy et al., 2015a; Reed et al., 2015; Gardner et al., 2015) where transformations are learned with strong coupling to the training images that are hard to generalize to novel ones. TFT is learned to perform three kinds of flow transformations (rotation, translation and scaling) with arbitrary transformation parameters. Moreover, it generalizes well to various datasets of different input dimensions, ranging from small patterns to natural images.

In the experiments, we train the proposed top-down flow transformer (TFT) on the MNIST dataset. We provide insights of our design choice. We demonstrate the results of the learned flow transformations by "inverting" transformed CNN features of images from several non-MNIST datasets. Comparing with the competing transformer (Reed et al., 2015) shows the immediate benefit our approach. We further train TFT on images synthesized from PASCAL VOC 2012 (Everingham et al.) and utilize it to perform network internal data augmentation to improve ImageNet (Russakovsky et al., 2015a) classification performance on VGG-16 (Simonyan & Zisserman, 2014).

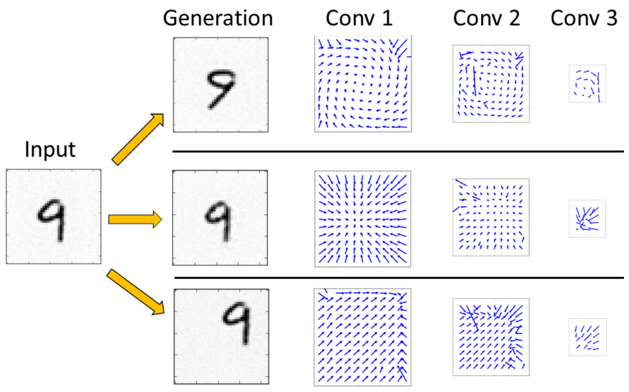


Figure 2. Feature flows of convolutional layers of a CNN after rotation ($-30°$), scaling ($\times 0.9$), and translation (move to top-right) estimated by TFT. Top-down spatial information is reflected in the flows, while deeper layers encode less spatial information.

2. Significance and Related Work

We first discuss the significance of our proposed top-down flow transformer (TFT).

Why a top-down generator? Top-down information can play a fundamental role in unraveling, understanding, and enriching the great representation power of deep convolutional neural networks to bring more transparency and interoperability. The feature flow maps learned by TFT on novel images show promise for the direction of top-down learning.

Why transform features across CNN layers? We are intrigued by the idea to understand how the CNN features change internally with respect to the spatial transformations. Capturing the generic feature flows under spatial transformations, our proposed TFT will aid in understanding the representation learned by a CNN in order to improve its robustness and to enrich its modeling and computing capabilities. This is not immediately available in the existing frameworks (Dosovitskiy et al., 2015a; Reed et al., 2015) where the models are heavily coupled with the specific training data.

Why not Spatial Transform Networks (STN) (Jaderberg et al., 2015)? In (Jaderberg et al., 2015), a spatial transformer was developed to explicitly account for the spatial manipulation of the data. This differentiable module can be inserted into existing CNNs, giving neural networks the ability to actively transform the image. However, the main goal of (Jaderberg et al., 2015) is to learn a differentiable transformation field through backpropagation to account for the spatial transformations using a localization network for classification to obtain the transformation parameters including rotation, scaling, translation, and sheering. In short STN learns to perform spatial transformation, **without controllable/user specified transformation parameters**, to match the output features whereas TFT studies the generator (with user specified transformation parameters) modeling the underlying changes to the feature maps in the result of spatial transformations.

Next, we discuss related work. We first compare TFT with three lines of work that are of high relevance.

1. Deep image analogy. The deep visual analogy-making work (Reed et al., 2015) shows impressive results to learn to transform and compose images. However, method like (Reed et al., 2015; Gardner et al., 2015) builds heavily on an encoder-decoder strategy that is strongly tied to the training images; it learns to output results in the image space without modeling the internal feature maps. Applying learned transformer (Reed et al., 2015) to novel images therefore leads to unsatisfactory results, as shown in the experiment section. Instead, our approach studies flow transformations on CNN features and generalizes well.

2. Learning image transformations. Learning to transform images has been a quite active research area recently. Existing methods that target on building transformation generators (Jaderberg et al., 2015; Lin & Lucey, 2016; Gregor et al., 2015; Kulkarni et al., 2015; Wu et al., 2017) trains CNN/RNN to perform transformation on the the output feature or the image space, which is different from our goal of studying the intrinsic flow transformations within the CNN networks. The key difference between TNT and STN (Jaderberg et al., 2015) has been discussed previously.

3. Feature transformation with SIFT-flow (Gallagher et al., 2015). An earlier attempt (Gallagher et al., 2015) (arXiv) has been made to study the feature layer deformation under explicit transformations based on flows computed from the SIFT-flow method (Liu et al., 2011). Although the work (Gallagher et al., 2015) has a similar big picture idea to our work here but it is very preliminary and builds transformation solely based on the SIFT-flow estimation (Liu et al., 2011). It therefore limited in several aspects: (1) only able to morph the features but cannot change the values; (2) may fail if SIFT-flow does not provide reliable estimation; (3) hard to generalize to arbitrary translation, rotation, and scaling. It relies heavily on the specially chosen parameters that do not work well under general situations. Our approach has a much greater learning capability than that in (Gallagher et al., 2015).

Next, we also discuss the existing literature in generative modeling. A family of mathematically well defined generators are defined in (Grenander, 1993) as the general pattern theory. Its algorithmic implementation however still needs a great deal of further development. Methods (Blake & Yuille, 1993; Cootes et al., 2001; Wu et al., 2010; Zhu et al., 2007) developed prior to the deep learning era are inspiring but they have limited modeling capabilities. Deep belief net (DBN) (Hinton et al., 2006) and generative adversarial networks (GAN) (Goodfellow et al., 2014) do not study the explicit top-down generator for the image transformation. Other generators that perform feed-forward mapping (Dosovitskiy et al., 2015a; Wu et al., 2016; Zhang et al., 2017) have transformations as input parameters but the process in (Dosovitskiy et al., 2015a) maps directly from the input parameter space to the output image space; it cannot be applied to generate novel categories and does not study the intrinsic transformation.

Existing works in flow estimation (Liu et al., 2011; Dosovitskiy et al., 2015b) are used to perform flow estimation, not as a generator for feature transformations. Our frameworks examine the transformations in CNN features that can be applied to generate novel images from various datasets and to perform data augmentation in a network internal fashion.

3. Top-down Flow Transformer

3.1. Architecture

The network comprises three layers: an aggregated feature transformation layer, a linear layer performing spatial transformation, and another aggregated feature transformation layer. Consider the feature maps $f_x \in \mathbb{R}^{n\times n\times m}$ of a convolutional layer with m channels from a CNN that is pretrained for a discriminative task (e.g., image classification) by feeding an image x. The aggregated flow transformation layer is given as:

$$\mathcal{F}(f_x) = f_x + \sum_{i=1}^{N} \mathcal{T}_i(f_x) - \sum_{i=N+1}^{2N} \mathcal{T}_i(f_x)$$

where N is an integer referred as the number of transformation functions for aggregation. We enforce N addition paths and N subtraction paths for the aggregation process. Each transformation function $\mathcal{T}_i(\cdot)$ is defined as a two-layer convolutional network, each layer has a convolution operation followed by the rectified linear (ReLU) activation. In our experiments demonstrated below, we use filter size 5x5, stride size 1 and number of filters in the first layer as either 16 or 32 (bottleneck type); we find $N = 4$ performs well.

The transformed feature maps are then fed into a linear layer that applies spatial transformations, including translation, rotation and scaling, to each channel individually (with bilinear interpolations). The specific spatial transformations

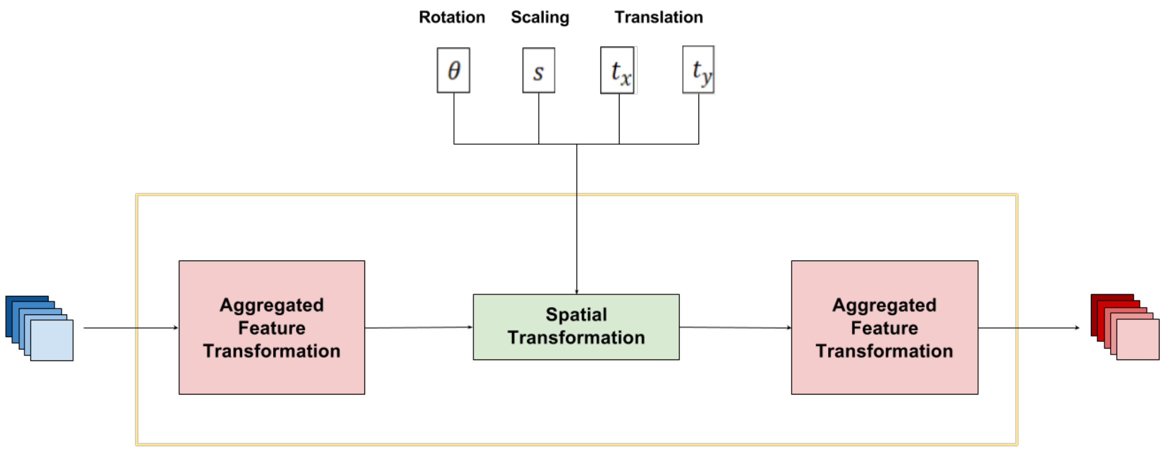


Figure 3. Architecture of our top-down flow transformer (TFT).

are modeled by the product of three transformation matrices: $M_{rot}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $M_{scale}(s) = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$, and $M_{tran}(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ where θ, s and (t_x, t_y) are top-down controls of rotation, scaling and translation, respectively. The results are further transformed by another aggregated residual layer to generate the output CNN features. Figure 2 illustrates the architecture. We build the TFT for each convolutional layer in the pre-trained CNN. For clarity, we denote the collection of the overall transformation parameters as $\Theta = (\theta, s, t_x, t_y)$.

3.2. Model Training

We train our proposed networks in a supervised manner by minimizing the average Euclidean distance $\mathcal{E} = \sum_{x,\Theta} E_\Theta(x)$ between the generated feature maps and the ground truth feature maps for any image x and any transformation parametrized by Θ in the training set. The ground truth features maps can be collected automatically from CNN features of input images that are under the corresponding spatial transformations. In specific, for an image x and its transformed version $\hat{x}$ under the transformation parametrized by Θ, $E_\Theta(x)$ is given as:

$$E_\Theta(x) = \|F_\Theta(f_x) - f_{\hat{x}}\|_2^2$$

where $F_\Theta(f_x)$ is the generated feature maps by our TFT and $f_{\hat{x}}$ is the ground truth feature maps from the input image $\hat{x}$.

3.3. Generating New Images

Upon obtaining the transformed feature maps $\hat{f}_x = F_\Theta(f_x)$ for some transformation parametrized by Θ, we can generate images by "inverting" them in CNNs, similar to the process in (Gatys et al., 2015), i.e., by minimizing the representation loss $E(x_{new}) = \left\|f_{x_{new}} - \hat{f}_x\right\|_2^2$ with respect to the generated image x_{new}, where $f_{x_{new}}$ is the CNN features of a given image x_{new}. Instead of picking up one convolutional layer, we use a set of layers altogether to generate images, with each layer transformed by a trained TFT. As our TFT can perform flow transformations to feature maps of all the convolutional layers while remaining consistent across them, we can minimize the representation loss across all layers simultaneously. One resulting benefit is better quality of generated images.

In practice, given a set of CNN layers $\{f_x(i)\}$, where i represents the i^{th} convolutional layer, we train the TFT for each one and generate images using a combination of the transformed feature maps $\{\hat{f}_x(i)\}$. We define the combined representation loss as:

$$E(x_{new}) = \sum_i \alpha_i \left\|f_{x_{new}}(i) - \hat{f}_x(i)\right\|_2^2 + \beta \cdot R_{TV}$$

where α_i and β are some hyper-parameters and $f_x(i)$ represents the feature maps from the i^{th} CNN layer of the input image x. We also add a regularization term R_{TV} defined as:

$$R_{TV} = \sum_{i,j} (x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2$$

similar to that in (Mahendran & Vedaldi, 2015).

3.4. Network Internal Data Augmentation

As suggested in (Gallagher et al., 2015) (arXiv), the learned generators can be applied to perform data augmentation inside the CNNs. Instead of feeding in newly generated images for CNN training, we can directly perform internal data augmentation in an on-line manner by applying learned

TFT to transform the CNN features. In our experiments, we perform fine-tuning on CNN via TFT trained upon the same CNN; by this means, the proposed TFT that learns to capture CNN feature flow under spatial transformation can be used to improve CNN's invariance to such spatial transformations. We examine this in the experiments section.

3.5. Flow Field Calculation

A feature map with just one black pixel and all other pixels white, referred as a unit feature map, is fed into the feature transformation model. The location of the black pixel is the starting point of a flow. The center of mass of the output feature map is calculated as the end point of the flow. Evenly spaced starting points are selected to calculate flows across a feature map space to generate a flow map of a certain transformation under a feature transformation model. Our TFT is shown to learn clear flow fields while achieving non-linear transformations across CNN channels.

4. Insights in Top-down Generator Design

Before the design of our proposed TFT, we come up with two other approaches, namely fully-connected model and gated affine model.

In the study of these approaches we gain insights in the feature transformation modeling. In the experiment of the fully-connected model, we find that patterns in deformed features are learnable, yet a highly non-linear model that does not explicitly capture the spatial information can fail to generalize to arbitrary top-down spatial transformations of the three kinds. In the experiment of the gated affine model, the networks explicitly utilize top-down spatial information via a parametrized gated affine transformation applied directly to the CNN features. However, because the model performs feature transformation in a highly linear channel-by-channel fashion, it lacks model complexity.

In the design of TFT, we combine the ideas and base it on the fact that, in a big picture, the deformation of feature maps resulted from spatial transformations resembles that of the input images, yet locally feature transformations are highly non-linear and occur across the channels. Accordingly, our proposed TFT is able to perform feature transformation with both strong interpretability (in terms of clear flow fields) and generalization (adequate model complexity with the explicit use of top-down spatial information).

4.1. Fully-connected Model

The fully-connected model consists of a two layer fully connected net, namely a parameter net that transforms top-down control parameters, and a 4 layer fully-connected net called a generator network which takes the concatenation of the transformed control parameters and CNN features from the source images and outputs the resulting feature maps. The architecture is illustrated in Figure 3.

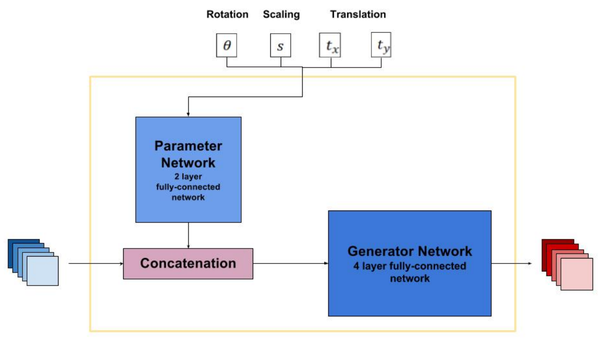


Figure 4. Architecture of the fully-connected model.

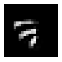
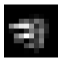
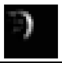
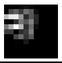
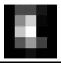
We train this model in the same setting as for TFT. Although it achieves good numerical and visual results of feature transformations on the MNIST dataset (the training dataset), it has two major drawbacks. First, it lacks the capability of explaining how feature maps are transformed in the sense of deformation fields. Second, it does not generalize well to arbitrary transformation parameters Θ outside of those used in the training process. We argue that this is due to the absence in the model of an explicit and direct usage of parameters that control the spatial transformations.

4.2. Gated Affine Model

Our other model is designed based on the assumption that the transformation of feature maps resembles that of the input images. For instance, if an image is rotated by 30 degrees, we assume the feature is also rotated, but not necessarily 30 degrees, as CNN feature extraction abstracts away some spatial information to achieve invariance. Accordingly, in our gated affine model we use gated transformation parameters and a distance matrix to re-parametrize the local feature manipulation in the context of affine transformation.

This gated affine model shows the promise of clear flow fields for feature transformations, yet it suffers from underfitting with rotation transformation. This results from the fact that this model performs the same transformation to each individual channel of the CNN features separately. As demonstrated in Table 1, lower level filters (such as edge detectors) each tends to only contain very limited spatial information (e.g., orientation and size) about the input images. Consequently, flow transformation modeling should consider all channels jointly for performing a nonlinear feature transformation given the arbitrary top-down control parameters.

Table 1. Feature map deformations resulted from rotation, translation, and scaling of the input image. conv1_1 and conv1_2 are two channels of the first conv layer; conv2_1 and conv2_2 similarly. All filters are taken from a CNN pre-trained on MNIST, which has 3 convolutional layers and 2 fully-connected layers.

	image	conv1_1	conv1_2	conv2_1	conv2_2	conv3_1
Original						
Rotation (-40°)						
Scaling (x0.8)						
Translation (up 8, left 8)						

4.3. Aggregated Flow Transformations

Our proposed TFT is inspired from both ideas. It utilizes a combination of CNN-based transformations and explicit spatial transformations across all feature channels. As a result, it generalizes well to several datasets without compromising the promise of clear feature flow fields. For the choice of specific transformation function, we find that aggregated residue transformation proposed in (Xie et al., 2017) particularly good for the task. We also find that using of subtraction path in addition generates better performance.

As TFT enforces a spatial transformation applied to all the channels, it also serves as a preliminary indicator of the extent to which the studied CNN preserves spatial information in each of its convolutional layer. For example, consider a CNN layer that contains all the spatial information of the input images, the corresponding TFT trained on this layer can theoretically perform well. Imagine a layer that is completely invariant to translation, rotation and scaling. It will be impossible for the TFT to work for that layer since the enforced spatial transformation will "ruin" the flow transformation; instead, an identity transformation is the ideal one. Our experiments show pertinent results as TFTs trained on layers deeper into the CNNs tend to perform worse.

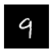
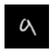
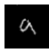
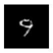
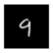
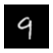
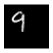
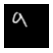
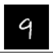
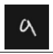
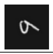
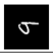
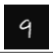
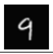
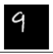
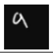
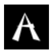
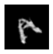
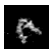
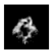
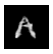
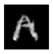
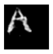
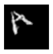
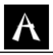
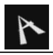
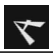
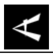
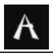
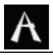
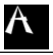
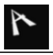
5. Experiments

We train our top-down flow transformer (TFT) on the MNIST dataset and evaluate the learned generator on images from several datasets, including MNIST, notMNIST, Kimia-99 (Belongie et al., 2002), MPEG-7 Shape (Latecki et al., 2000), and COIL-20 (Nene et al., 1996). Under all three studied transformations, namely rotation, translation and scaling, we generate new images from transformed feature maps by applying TFT. We compare our results to those of (Reed et al., 2015) (DVAM) on MNIST and notMNIST dataset. We achieve better results both graphically and numerically. Our framework is able to perform out-of-bound transformations, i.e., transformations with arbitrary parameters that are out of the range in the training process. Our model can generalize well to new datasets as the learned flow transformations are generic. Moreover, as TFT is not tied to fixed size of the input CNN features, it can perform flow transformations for arbitrary size of the input images, while existing method (Reed et al., 2015) cannot.

5.1. Training and Evaluation on MNIST

We resize each image of dimension 28 x 28 in the MNIST dataset to 44 x 44 by zero-padding the original images so that each image has enough space to perform translation and scaling. For data in the training set, we perform a combination of all three studied transformations to the input images and output the CNN feature maps across different convolutional layers. Specifically, for translation, we perform two dimensional shifting of the input images, with each axis ranging from +7 to -7 pixels, i.e., 225 combinations. For rotation, we perform with 13 different angles, namely rotate the input images by $5°$, $10°$, $15°$, $20°$, $25°$, $30°$ clockwise and counterclockwise as well as $0°$. For scaling, we choose three factors: 0.9, 1.0 and 1.1 (1.0 indicates no scaling). These four parameters are the top-down transformation parameters in the training set. We form 3-tuples $(f_{x_{ori}}, f_{x_\Theta}, \Theta)$, where $f_{x_{ori}}$ is the feature maps of the original images, f_{x_Θ} is the feature maps of the corresponding images after performing spatial transformation parameterized by Θ. Our training set is a collection of these 3-tuples. In practice, we apply the combinations of three transformations with randomly generated parameters from the range specified above.

Table 2. Comparison of transformations on MNIST (within dataset) and notMNIST dataset. Rotations of $60°$ and $90°$ is beyond training settings (from $-30°$ to $30°$). Images generated by DVAM on notMNIST is vague and even lost its pattern when signals are out-of-bound. Images generated by our model show clear pattern of transformation.

	Original Image	Rotation (30°)	Rotation (60°)	Rotation (90°)	Scaling (x0.9)	Scaling (x1.1)	Translation	Combination
DVAM								
Ours								
DVAM								
Ours								

In our experiments, we use a traditional CNN (with 3 convolutional layers, each is conv+relu+max-pooling and 2 fully connected layers) pre-trained on MNIST. We train three TFTs for the three convolutional layers, respectively, and use all these conv layers to generate new images.

The learning process of our TFT is supervised and the objective function is simply the Euclidean distance between the generated feature maps and the target ones, as mentioned in section 3.2. We train our networks by 200k steps with a L_2 regularizer of coefficient 0.0001. We use the ADAM optimizer (Kingma & Ba, 2014) with learning rate of 0.0001. We set the batch size to be 128.

We also train the networks proposed in (Reed et al., 2015), namely DVAM in the same settings by using the same training set. We compare our graphical results of generated images from MNIST to those of DVAM in Table 2. When evaluated on MNIST, our approach outperforms theirs in terms of both feature flow representation and out-of-bound transformations.

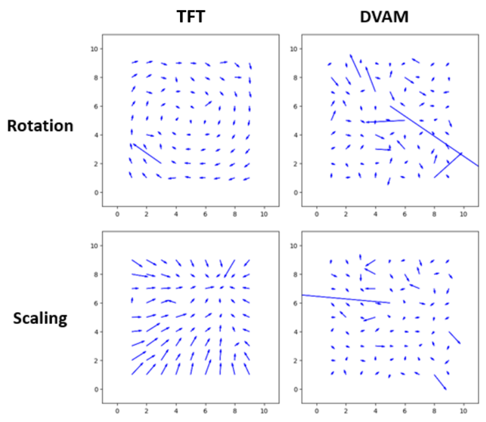


Figure 5. Feature flows of TFT and DVAM in (Reed et al., 2015). Flow generated by our model has clear pattern while the flow generated by DVAM does not.

5.1.1. Learned Feature Flow

One critical advantage of our top-down flow transformer is its clear representation of learned feature flow. By using the method described in section 3.5, we compute feature flow fields resulted from our proposed TFT learned from the CNN pre-trained on MNIST. As a comparison, we perform the counterpart from DVAM in (Reed et al., 2015), which has the similar CNN dimensions of the encoder-decoder structure. We perform this flow experiment on MNIST images with rotation and scaling. The learned flow transformations in our model has clearer rotation and scaling deformation patterns than that of (Reed et al., 2015), as illustrated by Figure 6.

Furthermore, within the same CNN, the feature flows of different convolution layers showcases a coherent representation of the top-down information ingested into our model. However, it is also evident that the top-down information plays a smaller role in the feature transformation of the higher layers as spatial information tends to be abstracted away there, namely in the third convolution layer of the CNN experimented.

5.1.2. Out-of-bound Transformations

Another advantage of our model is its robustness to out-of-bound parameters. We train our model on MNIST with rotation signal ranging from $-30°$ to $30°$, yet we test the rotation transformation with $60°$ and $90°$. In comparison, we also perform this experiment on (Reed et al., 2015). We can see that our model succeeds on the out-of-bound transformations while the model in (Reed et al., 2015) fails as illustrated in 3rd and 4th columns of Table 2.

5.2. Evaluation on non-MNIST Datasets

5.2.1. Evaluation on notMNIST Dataset

We further test our model and the analogy network (DVAM) in (Reed et al., 2015) trained on the same MNIST dataset to investigate their inter dataset performance. In this experiment, as well as all the other inter dataset experiments in section 5.2, we use our trained TFT based on the feature maps of the same traditional CNN, as discussed in section 5.1. To achieve similar visual and numerical effects to that of the MNIST dataset we normalize the notMNIST dataset using max norm. The results demonstrate that our networks have learned flow transformations of the CNN features that explicit utilize top-down information and generalize well to new types of data. Numerical data is in Table 4 and visual reconstructions of transformed features are illustrated in Table 2

5.2.2. Evaluation on Kimia-99, MPEG-7, and COIL-20 Dataset

Our model can be easily extended to feature maps of images with different sizes while (Reed et al., 2015) fails to do so. Our TFT has good performance in various other datasets, ranging from small patterns to real world images. We apply the same learned TFT to CNN features of images from Kimia-99 (Sharvit et al., 1998; Belongie et al., 2002), MPEG-7 Shape (Latecki et al., 2000) and COIL-20 (Nene et al., 1996) datasets. The generated images are illustrated in Table 5.

5.2.3. Evaluation on Natural Images

So far all our inter dataset experiments perform on grayscale images without background. We also apply our learned TFT for natural images. Since the pre-trained network based on MNIST only accepts single channel images, we perform flow transformation to the colored images channel by channel. We compare our results with that in (Gallagher et al., 2015), illustrated by Table 3. We use a mask on the input CNN features when applying our method.

Table 3. Comparison of transformations on images from a natural image.

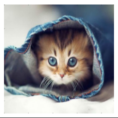
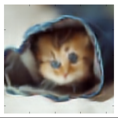
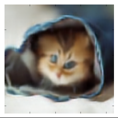
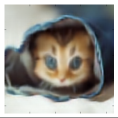
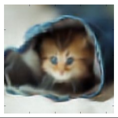
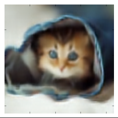
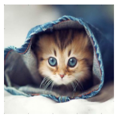
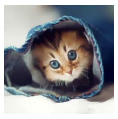
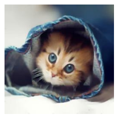
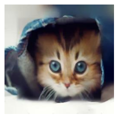
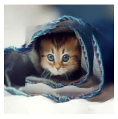
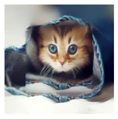
	Original	Rotation (30°)	Rotation (-30°)	Scaling (x1.3)	Scaling (x0.75)	Translation (up 30)
FlowPCA (Gallagher et al., 2015)						
Ours						

Table 4. Mean squared pixel prediction error according to different affine transformations of DVAM and our model on notMNIST

Model	translation	rotation	scaling	combination
DVAM	0.091315	0.077126	0.056829	0.062878
Ours	**0.001492**	**0.005311**	**0.004973**	**0.008571**

Table 5. Generated images by TFT from the Kimia (Sharvit et al., 1998), MPEG7 (Latecki et al., 2000), and COIL datasets (Nene et al., 1996). Images from different datasets are transformed using top-down transformers learned from the MNIST dataset. This is an inter dataset evaluation which demonstrates the effectiveness of our top-down model being generic and not tied to the training data.

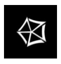
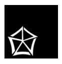
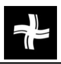
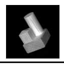
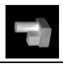
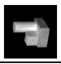
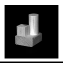
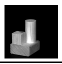
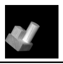
Input	Rotation			Scaling		Translation	Compositional
	30°	60°	90°	0.9	1.1	varied	varied
							
							
							

Table 6. ImageNet Validation Set Error (in %).

Method	top-1	top-5
VGG-16 (Simonyan & Zisserman, 2014)	24.8	7.5
VGG-16 after fine-tuning via TFT	24.4	7.3

5.3. Network Internal Data Augmentation

As previously mentioned, the learned TFT can further perform data augmentation inside CNNs. In this experiment, we train our proposed TFT on CNN features out of the third pooling layer from the pre-trained VGG-16 (Simonyan & Zisserman, 2014). We train the TFT using images synthesized from PASCAL VOC 2012 (Everingham et al.). In specific, we select 874 segmented images, each with single object approximately positioned in the center; we replace the background with the average color of images from the ImageNet dataset (Russakovsky et al., 2015a) and generate rotation, translation and scaling transformations, similar to the settings in the previous experiments with MNIST. After training the model, we insert the TFT into the corresponding layer of the pre-trained VGG-16 and perform fine-tuning. We sample the top-down spatial control parameters $\Theta = (\theta, s, t_x, t_y)$ uniformly from $[-30°, 30°] \times [0.8, 1.2] \times [-20, 20] \times [-20, 20]$ and accordingly achieve on-line data augmentation inside the networks. We report our results in Table 6. Our fine-tuning takes 3 epochs for the network internal data augmentation .

6. Conclusions

We have developed top-down feature transformer (TFT) that learns a top-down generator by studying the internal transformations across CNN layers. The learned transformer is illustrated on both within and across datasets which demonstrates its clear advantage over models that are heavily learned through data-driven techniques. TFT points to a promising direction within the study of a CNN's internal representation and top-down processes.

Acknowledgments This work is supported by NSF IIS-1618477 and NSF IIS-1717431.

References

Belongie, Serge, Malik, Jitendra, and Puzicha, Jan. Shape matching and object recognition using shape contexts. *TPAMI*, 24(4):509–522, 2002.

Blake, Andrew and Yuille, Alan. Active vision. 1993.

Cootes, Timothy F., Edwards, Gareth J., and Taylor, Christopher J. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

Dodwell, Peter C. The lie transformation group model of visual perception. *Perception & Psychophysics*, 34(1): 1–16, 1983.

Dosovitskiy, A., Springenberg, J. T., and Brox, T. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015a.

Dosovitskiy, Alexey, Fischer, Philipp, Ilg, Eddy, Hausser, Philip, Hazirbas, Caner, Golkov, Vladimir, van der Smagt, Patrick, Cremers, Daniel, and Brox, Thomas. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015b.

Elman, Jeffrey L. Distributed representations, simple recurrent networks, and grammatical. *Machine Learning*, 7: 195–225, 1991.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Gallagher, Patrick W, Tang, Shuai, and Tu, Zhuowen. What happened to my dog in that network: Unraveling top-down generators in convolutional neural networks. *arXiv preprint arXiv:1511.07125*, 2015.

Gardner, Jacob R, Upchurch, Paul, Kusner, Matt J, Li, Yixuan, Weinberger, Kilian Q, Bala, Kavita, and Hopcroft, John E. Deep manifold traversal: Changing labels with convolutional features. In *ECCV*, 2015.

Gatys, Leon A, Ecker, Alexander S, and Bethge, Matthias. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Gibson, James J. A theory of direct visual perception. *Vision and Mind: selected readings in the philosophy of perception*, pp. 77–90, 2002.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *NIPS*, 2014.

Gregor, Karol, Danihelka, Ivo, Graves, Alex, Rezende, Danilo, and Wierstra, Daan. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

Gregory, Richard Langton. *The intelligent eye*. Weidenfeld adn Nicolson, 1980.

Grenander, Ulf. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

Hill, Harold and Johnston, Alan. The hollow-face illusion: Object-specific knowledge, general assumptions or properties of the stimulus? 36:199–223, 01 2007.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural computation*, 18: 1527–1554, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 1997.

Jaderberg, Max, Simonyan, Karen, Zisserman, Andrew, et al. Spatial transformer networks. In *NIPS*, 2015.

Kersten, Daniel, Mamassian, Pascal, and Yuille, Alan. Object perception as bayesian inference. *Annual Review of Psychology*, 55(1):271–304, 2004.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

Kulkarni, Tejas D, Whitney, William F, Kohli, Pushmeet, and Tenenbaum, Josh. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pp. 2539–2547, 2015.

Latecki, Longin Jan, Lakamper, Rolf, and Eckhardt, T. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, volume 1, pp. 424–429, 2000.

Lin, Chen-Hsuan and Lucey, Simon. Inverse compositional spatial transformer networks. *arXiv preprint arXiv:1612.03897*, 2016.

Liu, C., Yuen, J., and Torralba, A. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *CVPR*, 2015.

Marr, David. *Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information*. MIT Press, 2010.

Nene, Sameer A., Nayar, Shree K., and Murase, Hiroshi. Columbia object image library (coil-20. Technical report, 1996.

Reed, Scott E, Zhang, Yi, Zhang, Yuting, and Lee, Honglak. Deep visual analogy-making. In *NIPS*. 2015.

Ridley Stroop, J. Studies of interference in serial verbal reactions. 121:15–23, 03 1992.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015a. doi: 10.1007/s11263-015-0816-y.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015b.

Sharvit, Daniel, Chan, Jacky, Tek, Huseyin, and Kimia, Benjamin B. Symmetry-based indexing of image databases. In *Content-Based Access of Image and Video Libraries, 1998. Proceedings. IEEE Workshop on*, pp. 56–62. IEEE, 1998.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Tu, Zhuowen, Chen, Xiangrong, Yuille, A. L., and Zhu, S. C. Image parsing: unifying segmentation, detection, and recognition. In *ICCV*, 2003.

Wu, Jiajun, Xue, Tianfan, Lim, Joseph J, Tian, Yuandong, Tenenbaum, Joshua B, Torralba, Antonio, and Freeman, William T. Single image 3d interpreter network. In *ECCV*, 2016.

Wu, Wanglong, Kan, Meina, Liu, Xin, Yang, Yi, Shan, Shiguang, and Chen, Xilin. Recursive spatial transformer (rest) for alignment-free face recognition. In *ICCV*, 2017.

Wu, Ying Nian, Si, Zhangzhang, Gong, Haifeng, and Zhu, Song-Chun. Learning active basis model for object detection and recognition. *International journal of computer vision*, 90(2):198–235, 2010.

Xie, Saining and Tu, Zhuowen. Holistically-nested edge detection. In *ICCV*, 2015.

Xie, Saining, Girshick, Ross, Dollár, Piotr, Tu, Zhuowen, and He, Kaiming. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995. IEEE, 2017.

Yuille, Alan L, Hallinan, Peter W, and Cohen, David S. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

Zhang, Quanshi, Cao, Ruiming, Wu, Ying Nian, and Zhu, Song-Chun. Growing interpretable part graphs on convnets via multi-shot learning. In *AAAI*, 2017.

Zhu, Song-Chun, Mumford, David, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.