Utilizando Spatial Transformer Networks

Lucas Caetano Possatti
Universidade Federal do Espírito Santo
Email: lucas.cpossatti@gmail.com

Rafael Horimoto de Freistas
Universidade Federal do Espírito Santo
Email: rafael.hdefreitas@gmail.com

***Resumo*—Redes Neurais Convolucionais (CNNs) estão sendo muito utilizadas para alcançar resultados do estado da arte na área de Visão Computacional. Uma propriedade desejável para tais modelos, que lidam com imagens, é serem capazes de desassociar posicionamento e deformações de objetos e de características da imagem. Assim sendo, tais modelos podem se tornar invariantes espacialmente ou até mesmo usarem a informação desassociada para ajudar na tarefa de inferência. Para alcansar tal propriedade, é explorado nesse trabalho, baseado no trabalho de Jaderberg et al [1], o uso de *Spatial Transformer Networks* (STNs). STNs são módulos que podem ser incluídos em outras redes neurais, provendo capacidades de transformação espacial, e podem ser usados juntamente com o algoritmo de backpropagation.**

I. Introdução

Redes Neurais Convolucionais (CNNs) estão sendo muito utilizadas para alcançar resultados do estado da arte na área de Visão Computacional. Uma propriedade desejável para tais modelos, que lidam com imagens, é serem capazes de desassociar posicionamento e deformações de objetos e de características da imagem. Assim sendo, tais modelos podem se tornar invariantes espacialmente ou até mesmo usarem a informação desassociada para ajudar na tarefa de inferência. Certa invariância espacial pode ser obtida através de camadas de *max-pooling* local. Porém, devido ao tipicamente pequeno espaço no qual o *max-pooling* atua, tal invariância só é alcansada após um profunda hierarquia de convoluções e camadas de *max-pooling*, de forma que as camadas intermediárias não são invariantes a largas transformações nos dados de entrada. Para alcansar tal propriedade, é explorado nesse trabalho, baseado no trabalho de Jaderberg et al [1], o uso de *Spatial Transformer Networks* (STNs). STNs são módulos que podem ser incluídos em outras redes neurais, provendo capacidades de transformação espacial, e podem ser usados juntamente com o algoritmo de backpropagation. A STN pode ter como entrada uma imagem ou um mapa de características, inferi uma transformação condicionada a entrada e aplica a transformação na entrada para obter a saída. A STN é dividida em três partes: rede de localização, gerador de malha e amostrador. A rede de localização aprende os parâmetros de uma transformação condicionados à entrada da STN. O gerador de malha então usa os parâmetros aprendidos pela rede de localização para transformar a malha de pixels da saída. A malha de pixels transformada defini as coordenadas correspondentes na entrada, porém tais coordenadas podem não ser inteiras, o que impossibilita definir diretamente cada pixel de saída em função de um único pixel de entrada. Para tratar tal problema, o amostrador usa as coordenadas da malha transformada juntamente com os pixels de entrada e aplica um algoritmo de mapeamento de textura para gerar os pixels de saída.

II. Trabalhos Correlatos

Segundo Jaderberg et al. [1], trabalhos anteriores modelam transformações com redes neurais [2], [3], [4], aprendem e analisam representações invariantes a transformações [5], [6], [7], [8], [9], [10] e usam mecanismos de atenção e detecção para seleção de características [11], [12], [13], [14], [15], [16]. Trabalho anterior por Hinton [2] visava atribuir frames canônicos de referência para partes de objetos, um tema que repercurtiu em [3] onde transformações afins 2D foram modeladas para criar um modelo generativo composto de partes transformadas. Os alvos do esquema de treino generativo são as imagens de entrada transformadas, com as transformações entre imagens de entrada e alvos dadas como entrada adicional para a rede. O resultado é um modelo generativo que consegue aprender a gerar imagens transformadas de objetos compondo partes. A noção de composição de partes transformadas é levada a frente por Tieleman [4], onde partes aprendidas são explicitamente transformadas por transformações afins, com a transformação inferida pela rede. Tais modelos generativos são capazes de aprender características discriminativas para classificação com supervisão de transformações. A invariância e equivariância de representações de CNNs em relação a transformações de imagens de entrada são estudadas em [9] estimando as relações lineares entre representações das imagens originais e transformadas. Cohen & Welling [6] analizam esse comportamento em relação a grupos simétricos, que também é explorado na arquitetura proposta por Gens & Domingos [7], resultando em mapas de características que são mais invariantes a grupos simétricos. Outras tentativas de criar representações invariantes a transformações são *scattering networks* [5], e CNNs que constroem bancos de filtros com filtros transformados [8], [10]. Stollenga et al. [17] usa uma política baseada em ativações de uma rede para reunir as respostas dos filtros da rede para uma subsequente passagem da mesma imagem e então permitir atenção para características específicas. Nesse trabalho, é focado alcansar representações invariantes manipulando os dados em vez dos extratores de características, algo que foi feito para agrupamento em [18]. Redes neurais com atenção seletiva manipulam os dados fazendo recortes, e então são capazes de aprender invariância a translação. Trabalhos como [11], [16] são treinados com

aprendizado de reforço para evitar a necessidade de um mecanismo de atenção diferenciável, enquanto [14] usa um mecanismo de atenção diferenciável usando kernels gaussianos em um modelo generativo. O trabalho por Girshick et al. [13] usa um algoritmo de proposição de região como uma forma de atenção, e [12] mostra que é possível fazer regressão de regiões salientes com uma CNN. O modelo apresentado nesse trabalho pode ser visto como uma generalização de atenção diferenciável para qualquer transformação espacial.

III. Metodologia

Para a realização dos experimentos utilizamos um computador com Ubuntu 14.04.4 LTS e três placas de vídeo GeForce GTX 660. O código para a realização dos experimentos pode ser encontrado no repositório deste projeto.

Neste trabalho nós realizamos alguns experimentos aplicando redes neurais em versões distorcidas do MNIST e do CIFAR-10. A versão distorcida do MNIST, nós chamamos de Cluttered MNIST (CM). E criamos imagens distorcidas do CIFAR-10 (C10) através de scripts próprios, e o chamamos de CIFAR-10-DISTORTED (C10D). A seguir, explicamos mais detalhadamente esses datasets.

A. Cluttered MNIST

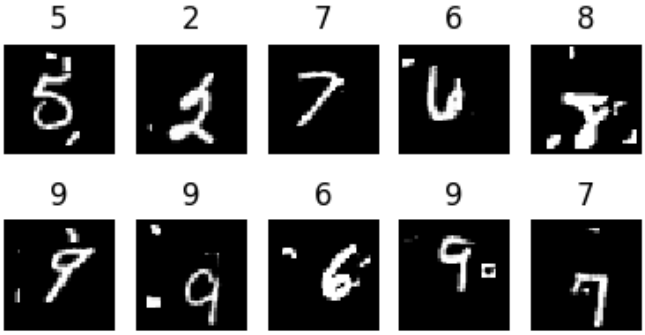

Figura 1. Exemplos de imagens do Cluttered MNIST. A classe de cada exemplo está acima do mesmo.

O Cluttered MNIST que usamos pode ser encontrado em [1]. O dataset foi construído utilizando as imagens do MNIST original, porém com translações aleatórias e adição de ruído (*clutter*) em cada imagem. Infelizmente não temos detalhes sobre os parâmetros usados para a criação desse dataset. Alguns exemplos de imagens do Cluttered MNIST são apresentadas na Figura 1. Neste dataset cada imagem está na resolução de 40×40 píxeis, com um único canal. E o dataset é dividido em 10000 imagens para treino; 1000 para validação; e 1000 para teste. Este é o mesmo dataset utilizado no experimento do modelo de pesquisa de STN do Tensorflow[2].

[1]https://github.com/daviddao/spatial-transformer-tensorflow/raw/master/data/mnist_sequence1_sample_5distortions5x5.npz

[2]https://github.com/tensorflow/models/tree/master/research/transformer

B. CIFAR-10-DISTORTED

O CIFAR-10[3] é um dataset que consiste de 60000 imagens RGB de 32×32 píxeis. Cada imagem pertencendo a uma de dez classes possíveis, com 6000 imagens para cada classe. O dataset está divido em 50000 imagens para treino e 10000 imagens para teste.

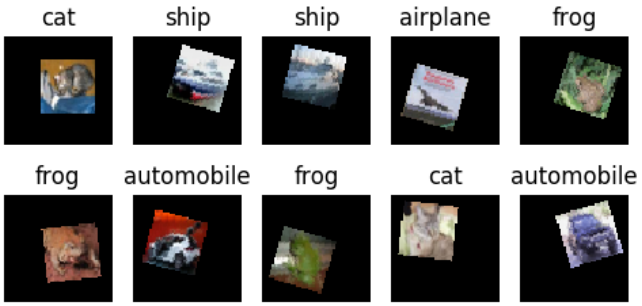

Figura 2. Exemplos de imagens do CIFAR-10-DISTORTED. A classe de cada exemplo está acima do mesmo.

Utilizando as imagens do CIFAR-10 nós criamos imagens distorcidas do mesmo ao aplicar rotação, e translação às imagens, visando criar um dataset de imagens distorcidas que chamamos de CIFAR-10-DISTORTED. O processo para a criação de uma imagem distorcida pode ser compreendido pelo seguinte processo: criamos um canvas preto 64×64 píxeis e colamos a imagem do CIFAR-10 (32×32 píxeis) no centro; aplicamos uma rotação aleatória no intervalo de $[-20, 20]$ graus; aplicamos uma translação aleatória em X e outra em Y no intervalo de $[-10, 10]$ píxeis. Alguns exemplos de imagens do CIFAR-10-DISTORTED são apresentados na Figura 2

Utilizando esse método criamos um *batch* de imagens de teste fixo, salvo em disco. Já para o treino, nós utilizamos o mesmo processo para gerar distorções aleatórias nas imagens de treino "*on the fly*".

A divisão entre imagens de treino e teste foi mantida. Ou seja, 50000 imagens de treino e 10000 imagens de teste. Porém, 10% das imagens de treino foram aleatoriamente selecionadas para compor um conjunto de validação, que não participou do treino (i.e. não foram utilizadas para atualizar os pesos da rede).

IV. Experimentos

Nós utilizamos dois modelos de redes neurais em nossos testes: Baseline e STN. Ambas são Redes Neurais Convolucionais (CNN), a única diferença entre os dois modelos está no fato de que a STN tem uma *Spatial Transformer Layer* como primeira camada da rede. Em todos os outros aspectos as redes são idênticas. A composição dessas duas redes variam para cada um dos datasets. A Tabela I mostra a composição da rede de localização (LocNet) utilizada no *Spatial Transformer* das STNs (em ambos os datasets). A Tabela II mostra a

[3]https://www.cs.toronto.edu/~kriz/cifar.html

composição do Baseline para o Cluttered MNIST. A Tabela III mostra a composição do Baseline para o CIFAR-10 e CIFAR-10-DISTORTED.

A notação que nós utilizamos para descrever as camadas das redes é a seguinte. Conv2D$[K, N, A]$ é uma camada convolucional 2D onde: K é o tamanho do kernel; N é o número de kernels; A é a função de ativação utilizada. FC$[N, A]$ é uma camada totalmente conectada, onde: N é o número de neurônios; A é a função de ativação. Em MaxPooling$[P]$, P é o tamanho do *cluster* do *pooling*.

LocNet
MaxPooling$[2 \times 2]$
Conv2D$[5 \times 5, 20, \text{Linear}]$
MaxPooling$[2 \times 2]$
Conv2D$[5 \times 5, 20, \text{Linear}]$
FC$[50, \text{ReLU}]$
FC$[6, \text{Linear}]$

Tabela I
COMPOSIÇÃO DO REDE DE LOCALIZAÇÃO USADA NAS STNS.

CM Baseline
Conv2D$[3 \times 3, 32, \text{ReLU}]$
MaxPooling$[2 \times 2]$
Conv2D$[3 \times 3, 32, \text{ReLU}]$
MaxPooling$[2 \times 2]$
FC$[256, \text{ReLU}]$
FC$[10, \text{Softmax}]$

Tabela II
COMPOSIÇÃO DO BASELINE USADO NO CLUTTERED MNIST (CM).

C10 Baseline
Conv2D$[3 \times 3, 48, \text{ReLU}]$
Conv2D$[3 \times 3, 48, \text{ReLU}]$
MaxPooling2D$[2 \times 2]$
Conv2D$[3 \times 3, 96, \text{ReLU}]$
Conv2D$[3 \times 3, 96, \text{ReLU}]$
MaxPooling2D$[2 \times 2]$
Conv2D$[3 \times 3, 192, \text{ReLU}]$
Conv2D$[3 \times 3, 192, \text{ReLU}]$
MaxPooling2D$[2 \times 2]$
FC$[512, \text{ReLU}]$
FC$[256, \text{ReLU}]$
FC$[10, \text{Softmax}]$

Tabela III
COMPOSIÇÃO DO BASELINE USADO NO CIFAR-10 (C10) E CIFAR-10-DISTORTED.

V. RESULTADOS

Nessa seção discutimos os resultados obtidos através da execução dos experimentos.

A. Resultados no Cluttered MNIST

A Figura 3 mostra a acurácia dos modelos ao longo do treino (10 épocas). É possível notar que os dois modelos alcançam acurácias de treinamento muito próximas, porém acurácias de validação bem distintas. Com a STN obtendo um desempenho melhor que o Baseline. Isso se confirma ao executarmos os

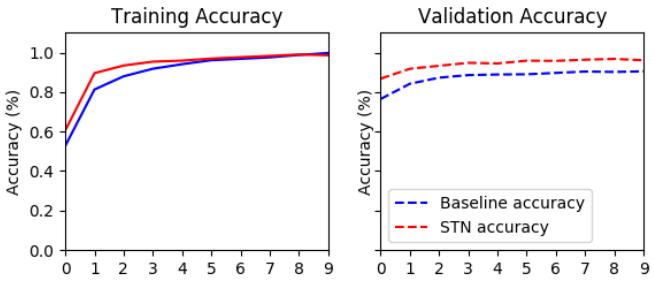

Figura 3. Acurácias de treino (esquerda) e validação (direita) ao longo do treinamento no Cluttered MNIST.

modelos no conjunto de teste, veja a Tabela IV. Observamos que a STN obteve 6 pontos percentuais a mais que o Baseline quanto a acurácia no conjunto de testes.

Modelo	**Acurácia**
Baseline	0.90
STN	0.96

Tabela IV
ACURÁCIA DOS MODELOS NAS IMAGENS DE TESTE DO CLUTTERED MNIST.

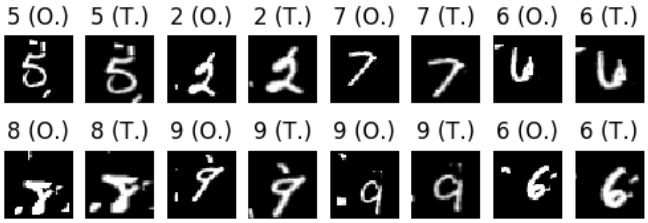

Figura 4. Exemplos de imagens transformadas pelo Spatial Transformer. Acima de cada imagem está a classificação correta, seguida de um marcador que diz se a imagem é original (O.) ou transformada (T.).

Na Figura 4 podemos ver alguns exemplos de imagens transformadas pela *Spatial Transformer* da STN. Na Figura, as imagens originais do dataset estão lado a lado com as suas respectivas versões transformadas. Observando as transformações, poderíamos dizer que a ST aprendeu a escalar e centralizar os dígitos de forma a apresentar o dígito de maneira mais normalizada para o restante da rede. Vemos também alguns exemplos de imagens que provavelmente se beneficiariam de rotação. Porém não notamos qualquer sinal de que a rede tenha aprendido a rotacionar os dígitos.

B. Resultados no CIFAR-10-DISTORTED

Além de executar os experimentos no CIFAR-10-DISTORTED, executamos também no CIFAR-10 (usando os mesmo modelos). A Figura 5 mostra a acurácia de treinamento e validação ao longo de 20 épocas de treino.

Podemos notar que após algumas épocas de treinamento a qualidade da STN piorou drasticamente. Investigando o problema, passamos algumas imagens do CIFAR-10-DISTORTED através do *Spatial Transformer* do modelo

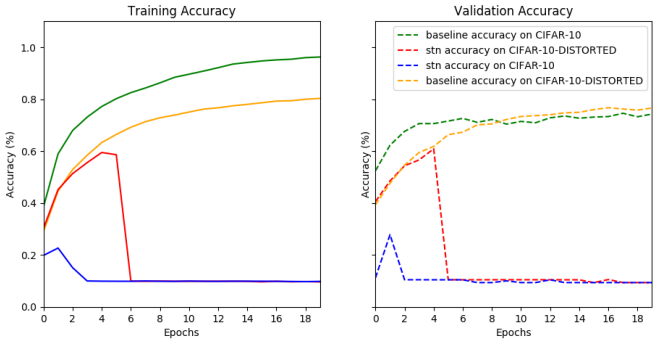

Figura 5. Acurácias de treino (esquerda) e validação (direita) ao longo do treinamento no CIFAR-10 e CIFAR-10-DISTORTED.

treinado, e obtivemos imagens completamente pretas. Isso nos leva a conjecturar que, após algum tempo de treinamento, a rede de localização se perde de forma que o foco do *Spatial Transformer* fica totalmente nas regiões pretas da figura e não consegue recuperar o foco na imagem colorida de forma alguma. Não entendemos exatamente porque isso aconteceu, mas pensamos que talvez poderíamos ter amenizado ou contornado esse problema adotando uma das seguintes estratégias: diminuindo a *Learning Rate*; utilizando um agendamento da *Learning Rate* (como é feito no artigo da STN); ou utilizando uma rede de localização mais robusta (mais parâmetros). Infelizmente não tivemos tempo para abordar essas possibilidades e realizar mais experimentos.

A Tabela V mostra a acurácia dos modelos treinados.

Dataset	**Modelo**	**Acurácia**
CIFAR-10	Baseline	0.74
CIFAR-10	STN	0.1
CIFAR-10-DISTORTED	Baseline	0.56
CIFAR-10-DISTORTED	STN	0.1

Tabela V
ACURÁCIA DOS MODELOS NO CONJUNTO DE TESTES DO CIFAR-10 E CIFAR-10-DISTORTED.

VI. CONCLUSÃO

O Spatial Transformer é um módulo construído para adicionar invariância espacial as CNNs. Ele faz isso ao aplicar uma transformação na imagem de entrada com objetivo de trazer foco ao objeto de interesse. Através dos experimentos que realizamos no *Cluttered MNIST* nós pudemos observar o funcionamento do ST e os resultados que ele pode trazer para o modelo. Realizamos também um segundo experimento nos datasets CIFAR-10 e CIFAR-10-DISTORTED, mas os modelos foram gravemente prejudicados pelo uso do ST, por motivos que não pudemos identificar por completo.

REFERÊNCIAS

[1] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015.

[2] G. F. Hinton, "A parallel computation that assigns canonical object-based frames of reference," in *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pp. 683–685, Morgan Kaufmann Publishers Inc., 1981.

[3] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, pp. 44–51, Springer, 2011.

[4] T. Tieleman, *Optimizing neural networks that generate images*. PhD thesis, University of Toronto (Canada), 2014.

[5] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[6] T. S. Cohen and M. Welling, "Transformation properties of learned visual representations," *arXiv preprint arXiv:1412.7659*, 2014.

[7] R. Gens and P. M. Domingos, "Deep symmetry networks," in *Advances in neural information processing systems*, pp. 2537–2545, 2014.

[8] A. Kanazawa, A. Sharma, and D. Jacobs, "Locally scale-invariant convolutional neural networks," *arXiv preprint arXiv:1412.5104*, 2014.

[9] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

[10] K. Sohn and H. Lee, "Learning invariant representations with local transformations," *arXiv preprint arXiv:1206.6418*, 2012.

[11] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[12] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154, 2014.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[14] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[15] J. Schmidhuber and R. Huber, "Learning to generate artificial fovea trajectories for target detection," *International Journal of Neural Systems*, vol. 2, no. 01n02, pp. 125–134, 1991.

[16] P. Sermanet, A. Frome, and E. Real, "Attention for fine-grained categorization," *arXiv preprint arXiv:1412.7054*, 2014.

[17] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Advances in Neural Information Processing Systems*, pp. 3545–3553, 2014.

[18] B. J. Frey and N. Jojic, "Fast, large-scale transformation-invariant clustering," in *Advances in neural information processing systems*, pp. 721–727, 2002.