Image Colorization using CNNs and Inception-Resnet-v2

Federico Baldassarre, Diego González Morín, Lucas Rodés-Guirao
{fedbal, diegogm, lucasrg} @kth.se

KTH Royal Institute of Technology

Abstract. In this report we review some of the most recent approaches to colorize gray-scale images using deep learning methods. Inspired by these, we propose a model which combines a deep Convolutional Neural Network trained from scratch with high-level features extracted from Inception-Resnet-v2 pre-trained model. Since our approach uses an encoder-decoder architecture purely based on CNNs it is able to process multiple sized images. Furthermore, we assess its "acceptance" by the public by means of a user study. Finally, we present a carousel of applications on different types of images, such as historical photographs.

Keywords: Colorization, CNN, Inception-ResNet-v2, Transfer Learning, Keras, TensorFlow

1 Introduction

Coloring gray-scale images can have a big impact in a wide variety of domains, for instance re-master of historical images and improvement of surveillance feeds. The information content of a gray-scale image is rather limited, thus adding the color components can provide more insights about its semantics. In the context of deep learning, models such as Inception [1], ResNet [2] or VGG [3] are usually trained using colored image datasets. When applying these networks on a gray-scale images, a prior colorization step can help improve the results. However, designing and implementing an effective and reliable system that automates this process still remains nowadays a challenging task. The difficulty increases even more if we aim at fooling the human eye.

In this regard, we propose a model that is able to colorize images to a certain extent, combining a deep Convolutional Neural Network architecture and the latest released Inception model to this date, namely Inception-ResNet-v2 [4], which is based on Inception v3 [1] and Microsoft's ResNet [2, 5]. While the deep CNN is trained from scratch, Inception-ResNet-v2 is used as a high-level feature extractor which provides information about the image contents that can help their colorization.

Due to time constraints, the size of the training dataset is considerably small, which leads to our model being restricted to a limited variety of images. Nevertheless, our results investigates some approaches carried out in other researches and validates the possibility to automate the colorization process.

1.1 Contribution

In brief, our contribution in this report can be summarized as follows.

- High-level feature extraction using a pre-trained model (Inception-ResNet-v2) to enhance the coloring process.
- Analysis and intuition behind a colorization architecture based on CNNs.
- Public acceptance evaluation by means of a user study.
- Colorization of historical pictures.

1.2 Organization

Section 2 briefly dives into the origins of image coloring techniques. Section 3 aims at presenting our approach and detailing its main components. Next, Section 4 presents our results, illustrating some colored images, and validates their "public acceptance" through a user study. Finally, Section 5 concludes the report with some notes on future work.

2 Background

In 2002, Welsh *et. al.* [6] presented a novel approach which was able to colorize an input image by transferring the color from a related reference image. Subsequent improvements of this method were proposed, exploiting low-level features [7] and introducing multi-modality on the pixel color values [8]. In parallel, another research line was initiated in 2004 by Levin *et. al.*, who proposed a scribble based method [9] which required the user to specify the colors of few image regions. This colorization methodology woke the interest of animators and cartoon-aimed techniques were proposed [10, 11]. The results from these approaches were certainly impressive at that time, however, the results were highly dependent on the artistic skills of the user. More recently, automatized approaches have been proposed. For instance, in [12] Desphande *et al.* conceived the coloring problem as a linear system problem.

In the last years, CNNs have been proven experimentally to almost halve the error rate for object recognition [13], which has led to a massive shift towards deep learning of the computer vision community. In this regard, Cheng Z. *et al.* [14] proposed a deep neural network using image descriptors (luminance, DAISY features [15] and semantic features) as inputs. In 2016, Iizuka, Serra *et. al.* [16] proposed a method based on using global-level and mid-level features to encode the images and colorize them. Our model draws its architecture on their approach and also serves as a validation. However, we introduce a pre-trained model into the equation. It is worth saying that similar approaches have been presented lately as well. For instance, Zhang *et. al.* [17] proposed a multi-modal scheme, where each pixel was given a probability value for each possible color. Another interesting approach was developed by Larsson *et al.* [18], in which a fully convolutional version of VGG-16 [19] with the classification layer discarded was used to build a color probability distribution for each pixel. In May 2017,

Zhang *et al.* [20] presented an an end-to-end CNN approach incorporating user "hints" in the spirit of scribble based methods, providing a color recommender system to help novice users and claiming to have enabled real-time use of their colorization system. These recent research, proves that this is an ongoing research line.

3 Approach

We consider images of size $H \times W$ in the CIE L*a*b* color space. [21]. Starting from the luminance component $\mathbf{X_L} \in \mathbb{R}^{H \times W \times 1}$, the purpose of our model is to estimate the remaining components to generate a fully colored version $\tilde{\mathbf{X}} \in \mathbb{R}^{H \times W \times 3}$. In short, we have a mapping $\mathcal{F}$ such that

$$\mathcal{F} : \mathbf{X_L} \rightarrow (\tilde{\mathbf{X}}_\mathbf{a}, \tilde{\mathbf{X}}_\mathbf{b}), \tag{1}$$

where $\tilde{\mathbf{X}}_\mathbf{a}$, $\tilde{\mathbf{X}}_\mathbf{b}$ are the a*, b* components of the reconstructed image, which combined with the input give the estimated colored image $\tilde{\mathbf{X}} = (\mathbf{X_L}, \tilde{\mathbf{X}}_\mathbf{a}, \tilde{\mathbf{X}}_\mathbf{b})$.

In order to be independent from the input size, our architecture is fully based on CNNs, a model that has been extensively studied and employed in the literature [22]. In brief, a convolutional layer is a set of small learnable filters that fit to specific local patterns in the input image. Layers close to the input look for simple patterns such as contours, while the ones closer to the output extract more complex features [23].

As already pointed out, we choose the CIE L*a*b* color space to represent the input images, since it separates the color characteristics from the luminance that contains the the main image features [24][25]. Combining the luminance with the predicted color components ensures a high level of detail on the final reconstructed image.

3.1 Architecture

Our model owes its architecture to [16]: given the luminance component of an image, the model estimates its a*b* components and combines them with the input to obtain the final estimate of the colored image. Instead of training a feature extraction branch from scratch, we make use of an Inception-ResNet-v2 network (referred to as Inception hereafter) and retrieve an embedding of the gray-scale image from its last layer. The network architecture we propose is illustrated in Fig. 1.

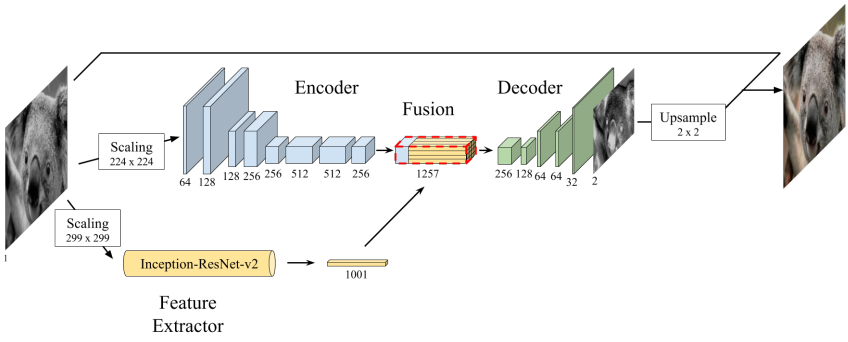

Fig. 1: An overview of the model architecture.

The network is logically divided into four main components. The encoding and the feature extraction components obtain mid and high-level features, respectively, which are then merged in the fusion layer. Finally, the decoder uses these features to estimate the output. Table 1 further details the network layers.

Table 1: Left: encoder network, mid: fusion network, right: decoder network. Each convolutional layer uses a ReLu activation function, except for the last one that employs an hyperbolic tangent function. Feature extraction details have been omitted, since it basically relies on Inception.

Layer	Kernels	Stride
conv	$64 \times (3 \times 3)$	2×2
conv	$128 \times (3 \times 3)$	1×1
conv	$128 \times (3 \times 3)$	2×2
conv	$256 \times (3 \times 3)$	1×1
conv	$256 \times (3 \times 3)$	2×2
conv	$512 \times (3 \times 3)$	1×1
conv	$512 \times (3 \times 3)$	1×1
conv	$256 \times (3 \times 3)$	1×1

Layer	Kernels	Stride
fusion	-	-
conv	$256 \times (1 \times 1)$	1×1

Layer	Kernels	Stride
conv	$256 \times (1 \times 1)$	1×1
conv	$128 \times (3 \times 3)$	1×1
upsamp	-	-
conv	$64 \times (3 \times 3)$	1×1
conv	$64 \times (3 \times 3)$	1×1
upsamp	-	-
conv	$32 \times (3 \times 3)$	1×1
conv	$2 \times (3 \times 3)$	1×1
upsamp	-	-

Preprocessing To ensure correct learning, the pixel values of all three image components are centered and scaled (according to their respective ranges [26]) in order to obtain values within the interval of $[-1, 1]$.

Encoder The Encoder processes $H \times W$ gray-scale images and outputs a $H/8 \times W/8 \times 512$ feature representation. To this end, it uses 8 convolutional layers with 3×3 kernels. Padding is used to preserve the layer's input size. Furthermore the first, third and fifth layers apply a stride of 2, consequentially halving the dimension of their output and hence reducing the number of computations required [27].

Feature Extractor High-level features, e.g. "underwater" or "indoor scene", convey image information that can be used in the colorization process. To extract an image embedding we used a pre-trained Inception model. First we scale the input image to 299×299. Next, we stack the image with itself to obtain a three-channel image (as shown in Fig. 2) in order to satisfy Inception's dimension requirements. Next, we feed the resulting image to the network and extract the output of the last layer before the softmax function. This results in a $1001 \times 1 \times 1$ embedding.

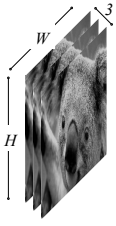

Fig. 2: Stacking the luminance component three times

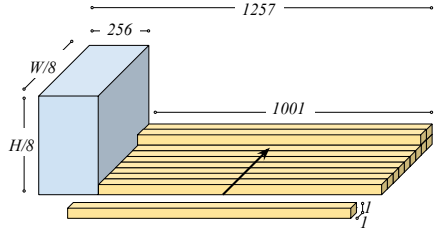

Fig. 3: Fusing the Inception embedding with the output of the convolutional layers of the encoder

Fusion The fusion layer takes the the feature vector from Inception, replicates it $HW/8^2$ times and attaches it to the feature volume outputted by the encoder along the depth axis. This method was introduced by [16] and is illustrated in Fig. 3. This approach, obtains a single volume with the encoded image and the mid-level features of shape $H/8 \times H/8 \times 1257$. By mirroring the feature vector and concatenating it several times we ensure that the semantic information conveyed by the feature vector is uniformly distributed among all spatial regions of the image. Moreover, this solution is also robust to arbitrary input image sizes, increasing the model flexibility. Finally, we apply 256 convolutional kernels of size 1×1, ultimately generating a feature volume of dimension $H/8 \times W/8 \times 256$.

Decoder Finally, the decoder takes this $H/8 \times W/8 \times 256$ volume and applies a series of convolutional and up-sampling layers in order to obtain a final layer with dimension $H \times W \times 2$. Up-sampling is performed using basic nearest neighbor approach so that the output's height and width are twice the input's.

3.2 Objective Function and Training

The optimal model parameters are found by minimizing an objective function defined over the estimated output and the target output. In order to quantify

the model loss, we employ the Mean Square Error between the estimated pixel colors in a*b* space and their real value. For a picture $\mathbf{X}$, the MSE is given by (2),

$$C(\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2HW} \sum_{k \in \{a,b\}} \sum_{i=1}^{H} \sum_{j=1}^{W} (X_{k_{i,j}} - \tilde{X}_{k_{i,j}})^2, \tag{2}$$

where $\boldsymbol{\theta}$ represents all model parameters, $X_{k_{i,j}}$ and $\tilde{X}_{k_{i,j}}$ denote the ij:th pixel value of the k:th component of the target and reconstructed image, respectively. This can easily be extended to a batch $\mathcal{B}$ by averaging the cost among all images in the batch, i.e. $1/|\mathcal{B}| \sum_{\mathbf{X} \in \mathcal{B}} C(\mathbf{X}, \boldsymbol{\theta})$.

While training, this loss is back propagated to update the model parameters $\boldsymbol{\theta}$ using Adam Optimizer [28] with an initial learning rate $\eta = 0.001$. During training we impose a fixed input image size to allow for batch processing. However, in later usage of the model, the size can be arbitrary.

4 Experiments and Discussion

As important as the network's architecture itself is the choice of the dataset. In the majority of the approaches to automatic image recoloring so far ImageNet has been extensively used [17][18]. Besides, ImageNet's impressive size (more than 14,000,000 images), extensive documentation and free access makes it appealing for our purpose. The dataset is composed of millions of pictures within a wide variety of sets. In particular, it based on the *name* nodes contained int the word dataset WordNet In order to simplify training and reduce running times, only a small subset of approx. 60,000 images is used.

ImageNet pictures are heterogeneous in shape, therefore all images in the training set are rescaled to 224×224 for the encoding branch input and to 299×299 for Inception. Each image gets stretched or shrunk as needed, but its aspect ratio is preserved by adding a white padding if needed.

4.1 Training

Of the approx 60,000 original images we held out the 10% to be used as validation data during training. The results presented in this report are drawn from this validation set and therefore the network never had the chance to see those images during training Adam optimizer was used during approximately 23 hours of training.

Complete details about the architecture, the image processing pipeline and our implementation in Keras [29] and TensorFlow [30] can be found in the project webpage[1].

The network was trained and tested using the Tegner nodes of The PDC Center for High Performance Computing at the KTH Royal Institute of Technology, leveraging the the NVIDIA CUDA Toolkit [31] and the NVIDIA Tesla

[1] https://github.com/baldassarreFe/deep-koalarization/

K80 Accelerator GPU to speed up the computationst. A batch size of 100 ruled out the risk of overflowing the GPU memory.

4.2 Results

Once trained, we fed our network with some images. The results turned out to be quite good for some of the images, generating near-photorealistic pictures. However, due to the small size of our training set our network performs better when certain image features appear. For instance, nature elements such as the sea or vegetation seem to be well recognized. However, specific objects are not always well colored. Fig. 4 illustrates results for some examples where our network produces alternative colored estimates.

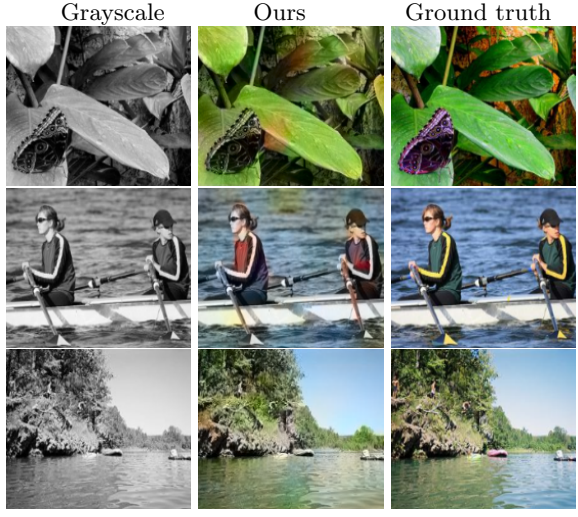

Fig. 4: In the first row our approach is capable of recognizing the green vegetation. However, the butterfly was not colored. Furthermore, in the example in the second row we observe that the network changes the color of the rowers' clothes from green-yellow to red-blue. Last row shows a landscape example where our model provided a photorealistic image.

Fig. 5 exposes generated color images using our method along with other state of the art approaches. Larsson *et al.*, Zhang *et al.* and we used ImageNet training set. Iizuka *et al.*, instead, used Places training dataset. Furthermore, we use the same objective function as Iizuka *et al.* (MSE loss). On the contrary, Larsson *et al.* and Zhang *et al.* use an un-rebalanced and rebalanced classification loss, respectively. From the results, we observed that although some results were quite good, some generated pictures tend to be low saturated, with the network producing a gray-ish color where the original would be brighter (e.g. with images of fruit, flowers or clothes). Our interpretation is that the network,

in its attempt to minimize the loss between images where e.g. flowers are red and others where flowers are blue, ends up doing very doing conservative predictions, namely assigning a neutral gray color.

Fig. 5: Comparison of the results obtained from our colorization network with other approaches. The first column shows the gray-scale input image. Columns 2-4 show the results of the automatic colorization models from Iizuka *et al.*, Larsson *et al.* and Zhang *et al.* (2016), respectively. Column 5 shows our results and, finally, the last column provides the corresponding ground truth images. In the presented examples, in rows 5 and 7 our method outperforms the other methods, generating more photo-realistic images. In the remaining images, some regions of the the generated images by our method lack of saturation. Images are from the ImageNet dataset (Russakovsky et al. 2015).

4.3 User Study

Although we can empirically obtain a measure of the performance of our model by using (2), we are also interested in how compelling the colors look to a human observer, which can be difficult to assess using solely mathematical tools. Thus, decided to evaluate the appearance of some artificially recolored images by means of a user study[2]. To this end, we chose twelve images, nine of which were recolored and are shown in Fig. 6, and asked, for each of them, the question "'Fake or real?". In doing this we picked our best results, discarding all the images that were not well recolored only selecting those that could "fool" the human eye. The poll was taken by 41 different users.

Fig. 6: For each recolored image we give the percentage of users that answered "real" to the question *Fake or real?* The images are sorted according to their "fooling capacity".

These results totally overcame our expectations. In particular, we can observe that in some cases the real-perception achieved almost 80.0 %. However, this results also indicate that credibility of the image strongly depends on what the image is portraying. We believe that incrementing the size and variability on the training set could partially mitigate this. Overall, we computed that 45.87% of the users miss-classified recolored images as originals. However, bear in mind that the recolored images for the user study were carefully selected from our best results.

[2] https://goo.gl/forms/nxPJUXhmZkeLYmsQ2

4.4 Historical Photographs

Fascinated by the power of this technique, we tested our model on historical pictures. The results are shown in Fig7. Since no ground truth exists, the results are only interpretable by means of personal judgment.

Fig. 7: Example of recolored historical images, from left to right: Titanic, before the iceberg(1912),The 1927 Solvay Conference in Brussels(1927), Hindenburg disaster (1937), The Great Dictator (Chaplin, 1940), Queen Elizabeth(1969)

5 Conclusions and Future Work

This project validates that an end-to-end deep learning architecture is suitable for image colorization tasks. Our approach is able to successfully color high-level image components such as the sky, the sea or the forest. On the other hand, the performances in coloring the smallest details is still to be improved. As we only used a reduced subset of ImageNet, only a small portion of the spectrum of possible subjects is represented, therefore, the performance on unseen images highly depends on their specific contents. To overcome this issue, our network should be trained over a larger training dataset.

In this regard, a probabilistic approach in the spirit of [17] seems more adequate. We believe that a better mapping between luminance and a*b* components could be achieved by an approach similar variational autoencoders, also allowing to generate images by sampling from a probability distribution.

Finally, it could be interesting to apply colorization techniques to video sequences, which could potentially re-master old documentaries. This of course would require adapting the network architecture to accommodate temporal coherence between subsequent frames.

Overall, we believe that end to end image translation and colorization has a huge potential in the future and will eventually reduce hours of supervised work.

References

1. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR **abs/1512.00567** (2015)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015)
3. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
4. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR **abs/1602.07261** (2016)
5. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. CoRR **abs/1603.05027** (2016)
6. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. In: ACM Transactions on Graphics (TOG). Volume 21., ACM (2002) 277–280
7. Ironi, R., Cohen-Or, D., Lischinski, D.: Colorization by example. In: Rendering Techniques, Citeseer (2005) 201–210
8. Charpiat, G., Hofmann, M., Schölkopf, B.: Automatic image colorization via multimodal predictions. Computer Vision–ECCV 2008 (2008) 126–139
9. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: ACM Transactions on Graphics (ToG). Volume 23., ACM (2004) 689–694
10. Qu, Y., Wong, T.T., Heng, P.A.: Manga colorization. In: ACM Transactions on Graphics (TOG). Volume 25., ACM (2006) 1214–1220
11. Sỳkora, D., Dingliana, J., Collins, S.: Lazybrush: Flexible painting tool for hand-drawn cartoons. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 599–608
12. Deshpande, A., Rock, J., Forsyth, D.: Learning large-scale automatic image colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 567–575
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
14. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423
15. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE (2008) 1–8
16. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics (TOG) **35**(4) (2016) 110
17. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European Conference on Computer Vision, Springer (2016) 649–666
18. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: European Conference on Computer Vision, Springer (2016) 577–593
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
20. Zhang, R., Zhu, J.Y., Isola, P., Geng, X., Lin, A.S., Yu, T., Efros, A.A.: Real-time user-guided image colorization with learned deep priors. arXiv preprint arXiv:1705.02999 (2017)

21. Robertson, A.R.: The cie 1976 color-difference formulae. Color Research & Application **2**(1) (1977) 7–11
22. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016) http://www.deeplearningbook.org.
23. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision, Springer (2014) 818–833
24. Bora, D.J., Kumar Gupta, A., Ahmad Fayaz, K.: Unsupervised diverse colorization via generative adversarial networks. International Journal of Emerging Technology and Advanced Engineering (2015)
25. Nixon, M.S., Aguado, A.S.: Feature Extraction & Image Processing for Computer Vision. Elsevier Ltd (2002)
26. Hoffman, G.: Cielab colorspace. Technical report, University of Applied Sciences, Emden (Germany), http://docs-hoffmann.de/cielab03022003.pdf (2003)
27. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. CoRR **abs/1412.6806** (2014)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
29. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
30. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.
31. Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming with cuda. Queue **6**(2) (March 2008) 40–53