Technical Writing:
The Anatomy of a Research Paper

Preethi Jyothi
Department of CSE, IIT Bombay

March 13, 2024

Disclaimer

- A talk about technical writing can never be complete

- There are a large number of excellent online resources on this topic

  ✓ [Knuth89] “Mathematical Writing”, Donald E. Knuth, Tracy Larrabee and Paul M. Roberts, 1989

  ✓ [Gopen90] “The Science of Scientific Writing”, George Gopen and Judith Swan, American Scientist, 1990

  - [Lisberger11] “From Science to Citation: How to Publish a Successful Scientific Paper”, Stephen Lisberger, 2011

  ✓ [Mensh17] “Ten Simple Rules for Structuring Papers”, Brett Mensh and Konrad Kording, PLOS Computational Biology, 2017

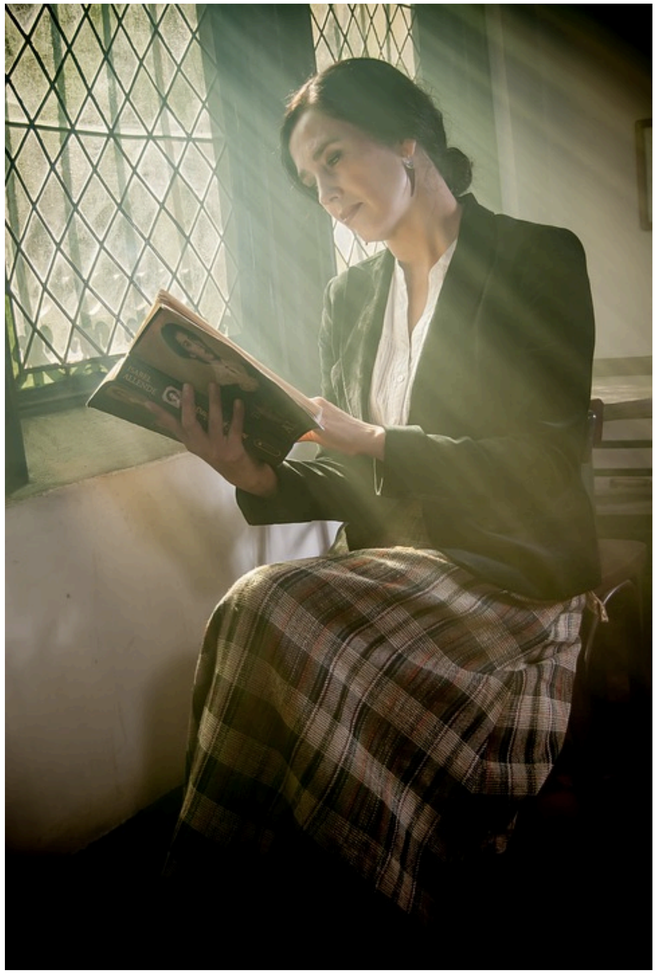
  ✓ [Freeman18], “How to Write a Good Research Paper”, Bill Freeman, Good Citizen of CVPR, 2018 (https://www.cc.gatech.edu/~parikh/citizenofcvpr/)

This talk borrows content from the references marked with ✓

Easy to lose sight of

> The fundamental purpose of scientific discourse is not the mere presentation of information and thought, but rather its actual communication. It ***does not matter how pleased an author might be to have converted all the right data into sentences and paragraphs;*** it matters only ***whether a large majority of the reading audience accurately perceives what the author had in mind.***
>
> *George Gopen, Judith Swan*

For a reviewer of your paper



it’s less like this…

and more like this!

Image from: http://phdcomics.com/comics.php?f=980

Skeleton of a paper [Mensh17]

- **Context**: State the problem you are tackling. Tell the reader why he/she should care about it.

- **Content**: What is your solution to the problem? Why is it different from previously proposed solutions?

- **Conclusion**: Discuss your approach. Analyse its benefits and weaknesses.

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Pick an informative and non-generic title.
Should reflect the main ideas in the paper.
Include prominent keywords that will help in better indexing.

Convey the entire message of the paper in the abstract.
Maintain the “context, content, conclusion” structure here.

Can sell or sink your paper. Spend considerable time on this section!
Explain gaps in the literature and how your paper fills the gap.
Keep the reader interested and clearly summarise your main results.

Important to give proper credit. Be generous.
Relate your ideas well to what was previously proposed.

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Organize your main content well. Typically multiple sections.
Ensure logical flow across and within sections.
Figures are a good idea.

Results to convince the reader that the central claim is justified.
Figures are very important. Also, captions of the figures!

Evaluate strengths/weaknesses of your approach.
Preempt reviewers' comments about shortcomings.

Example paper from [Freeman18]

Removing Camera Shake from a Single Photograph

Rob Fergus[1] Barun Singh[1] Aaron Hertzmann[2] Sam T. Roweis[2] William T. Freeman[1]

[1]MIT CSAIL [2]University of Toronto

Figure 1: *Left*: An image spoiled by camera shake. *Middle*: result from Photoshop "unsharp mask". *Right*: result from our algorithm.

Abstract

Camera shake during exposure leads to objectionable image blur and ruins many photographs. Conventional blind deconvolution methods typically assume frequency-domain constraints on images, or overly simplified parametric forms for the motion path during camera shake. Real camera motions can follow convoluted paths, and a spatial domain prior can better maintain visually salient image characteristics. We introduce a method to remove the effects of camera shake from seriously blurred images. The method assumes a uniform camera blur over the image and negligible in-plane camera rotation. In order to estimate the blur from the camera shake, the user must specify an image region without saturation effects. We show results for a variety of digital photographs taken from personal photo collections.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement, G.3 [Artificial Intelligence]: Learning

Keywords: camera shake, blind image deconvolution, variational learning, natural image statistics

1 Introduction

Camera shake, in which an unsteady camera causes blurry photographs, is a chronic problem for photographers. The explosion of consumer digital photography has made camera shake very prominent, particularly with the popularity of small, high-resolution cameras whose light weight can make them difficult to hold sufficiently steady. Many photographs capture ephemeral moments that cannot be recaptured under controlled conditions or repeated with different camera settings — if camera shake occurs in the image for any depth-of-field. A tripod, or other specialized hardware, can eliminate camera shake, but these are bulky and most consumer photographs are taken with a conventional, handheld camera. Users may avoid the use of flash due to the unnatural tonescales that result. In our experience, many of the otherwise favorite photographs of amateur photographers are spoiled by camera shake. A method to remove that motion blur from a captured photograph would be an important asset for digital photography.

Camera shake can be modeled as a blur kernel, describing the camera motion during exposure, convolved with the image intensities. Removing the unknown camera shake is thus a form of blind image deconvolution, which is a problem with a long history in the image and signal processing literature. In the most basic formulation, the problem is underconstrained: there are simply more unknowns (the original image and the blur kernel) than measurements (the observed image). Hence, all practical solutions must make strong prior assumptions about the blur kernel, about the image to be recovered, or both. Traditional signal processing formulations of the problem usually make only very general assumptions in the form of frequency-domain power laws; the resulting algorithms can typically handle only very small blurs and not the complicated blur kernels often associated with camera shake. Furthermore, algorithms exploiting image priors specified in the frequency domain may not preserve important spatial-domain structures such as edges.

This paper introduces a new technique for removing the effects of unknown camera shake from an image. This advance results from two key improvements over previous work. First, we exploit recent research in natural image statistics, which shows that photographs of natural scenes typically obey very specific distributions of image gradients. Second, we build on work by Miskin and MacKay [2000], adopting a Bayesian approach that takes into account uncer-

Removing Camera Shake from a Single Photograph
Fergus et al. 2006

above assumptions are violated; however, they may be acceptable to consumers in some cases, and a professional designer could touch-up the results. In contrast, the original images are typically unusable, beyond touching-up — in effect our method can help "rescue" shots that would have otherwise been completely lost.

2 Related Work

The task of deblurring an image is image deconvolution; if the blur kernel is not known, then the problem is said to be "blind". For a survey on the extensive literature in this area, see [Kundur and Hatzinakos 1996]. Existing blind deconvolution methods typically assume that the blur kernel has a simple parametric form, such as a Gaussian or low-frequency Fourier components. However, as illustrated by our examples, the blur kernels induced during camera shake do not have simple forms, and often contain very sharp edges. Similar low-frequency assumptions are typically made for the input image, e.g., applying a quadratic regularization. Such assumptions can prevent high frequencies (such as edges) from appearing in the reconstruction. Caron et al. [2002] assume a power-law distribution on the image frequencies; power-laws are a simple form of natural image statistics that do not preserve local structure. Some methods [Jalobeanu et al. 2002; Neelamani et al. 2004] combine power-laws with wavelet domain constraints but do not work for the complex blur kernels in our examples.

Deconvolution methods have been developed for astronomical images [Gull 1998; Richardson 1972; Tsumuraya et al. 1994; Zarowin 1994], which have statistics quite different from the natural scenes we address in this paper. Performing blind deconvolution in this domain is usually straightforward, as the blurry image of an isolated star reveals the point-spread-function.

Another approach is to assume that there are multiple images available of the same scene [Bascle et al. 1996; Rav-Acha and Peleg 2005]. Hardware approaches include: optically stabilized lenses [Canon Inc. 2006], specially designed CMOS sensors [Liu and Gamal 2001], and hybrid imaging systems [Ben-Ezra and Nayar 2004]. Since we would like our method to work with existing cameras and imagery and to work for as many situations as possible, we do not assume that any such hardware or extra imagery is available.

Recent work in computer vision has shown the usefulness of heavy-tailed natural image priors in a variety of applications, including denoising [Roth and Black 2005], superresolution [Tappen et al. 2003], intrinsic images [Weiss 2001], video matting [Apostoloff and Fitzgibbon 2005], inpainting [Levin et al. 2003], and separating reflections [Levin and Weiss 2004]. Each of these methods is effectively "non-blind", in that the image formation process (e.g., the blur kernel in superresolution) is assumed to be known in advance.

Miskin and MacKay [2000] perform blind deconvolution on line art images using a prior on raw pixel intensities. Results are shown for small amounts of synthesized image blur. We apply a similar varia-

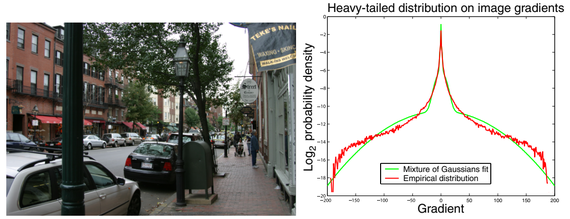


Figure 2: *Left:* A natural scene. *Right:* The distribution of gradient magnitudes within the scene are shown in red. The y-axis has a logarithmic scale to show the heavy tails of the distribution. The mixture of Gaussians approximation used in our experiments is shown in green.

the sensor irradiance. The latent image $\mathbf{L}$ represents the image we would have captured if the camera had remained perfectly still; our goal is to recover $\mathbf{L}$ from $\mathbf{B}$ without specific knowledge of $\mathbf{K}$.

In order to estimate the latent image from such limited measurements, it is essential to have some notion of which images are *a-priori* more likely. Fortunately, recent research in natural image statistics have shown that, although images of real-world scenes vary greatly in their absolute color distributions, they obey *heavy-tailed* distributions in their gradients [Field 1994]: the distribution of gradients has most of its mass on small values but gives significantly more probability to large values than a Gaussian distribution. This corresponds to the intuition that images often contain large sections of constant intensity or gentle intensity gradient interrupted by occasional large changes at edges or occlusion boundaries. For example, Figure 2 shows a natural image and a histogram of its gradient magnitudes. The distribution shows that the image contains primarily small or zero gradients, but a few gradients have large magnitudes. Recent image processing methods based on heavy-tailed distributions give state-of-the-art results in image denoising [Roth and Black 2005; Simoncelli 2005] and superresolution [Tappen et al. 2003]. In contrast, methods based on Gaussian prior distributions (including methods that use quadratic regularizers) produce overly smooth images.

We represent the distribution over gradient magnitudes with a zero-mean mixture-of-Gaussians model, as illustrated in Figure 2. This representation was chosen because it can provide a good approximation to the empirical distribution, while allowing a tractable estimation procedure for our algorithm.

4 Algorithm

There are two main steps to our approach. First, the blur kernel is estimated from the input image. The estimation process is performed in a coarse-to-fine fashion in order to avoid local minima. Second, using the estimated kernel, we apply a standard deconvo-

4.1 Estimating the blur kernel

Given the grayscale blurred patch $\mathbf{P}$, we estimate $\mathbf{K}$ and the latent patch image $\mathbf{L}_p$ by finding the values with highest probability, guided by a prior on the statistics of $\mathbf{L}$. Since these statistics are based on the image gradients rather than the intensities, we perform the optimization in the gradient domain, using $\nabla\mathbf{L}_p$ and $\nabla\mathbf{P}$, the gradients of $\mathbf{L}_p$ and $\mathbf{P}$. Because convolution is a linear operation, the patch gradients $\nabla\mathbf{P}$ should be equal to the convolution of the latent gradients and the kernel: $\nabla\mathbf{P} = \nabla\mathbf{L}_p \otimes \mathbf{K}$, plus noise. We assume that this noise is Gaussian with variance σ^2.

As discussed in the previous section, the prior $p(\nabla\mathbf{L}_p)$ on the latent image gradients is a mixture of C zero-mean Gaussians (with variance v_c and weight π_c for the c-th Gaussian). We use a sparsity prior $p(\mathbf{K})$ for the kernel that encourages zero values in the kernel, and requires all entries to be positive. Specifically, the prior on kernel values is a mixture of D exponential distributions (with scale factors λ_d and weights π_d for the d-th component).

Given the measured image gradients $\nabla\mathbf{P}$, we can write the posterior distribution over the unknowns with Bayes' Rule:

$$p(\mathbf{K}, \nabla\mathbf{L}_p|\nabla\mathbf{P}) \propto p(\nabla\mathbf{P}|\mathbf{K}, \nabla\mathbf{L}_p)p(\nabla\mathbf{L}_p)p(\mathbf{K}) \quad (2)$$

$$= \prod_i \mathtt{N}(\nabla\mathbf{P}(i)|(\mathbf{K} \otimes \nabla\mathbf{L}_p(i)), \sigma^2) \quad (3)$$

$$\prod_i \sum_{c=1}^{C} \pi_c \mathtt{N}(\nabla\mathbf{L}_p(i)|0, v_c) \prod_j \sum_{d=1}^{D} \pi_d \mathtt{E}(\mathbf{K}_j|\lambda_d)$$

where i indexes over image pixels and j indexes over blur kernel elements. $\mathtt{N}$ and $\mathtt{E}$ denote Gaussian and Exponential distributions respectively. For tractability, we assume that the gradients in $\nabla\mathbf{P}$ are independent of each other, as are the elements in $\nabla\mathbf{L}_p$ and $\mathbf{K}$.

A straightforward approach to deconvolution is to solve for the maximum a-posteriori (MAP) solution, which finds the kernel $\mathbf{K}$ and latent image gradients $\nabla\mathbf{L}$ that maximizes $p(\mathbf{K}, \nabla\mathbf{L}_p|\nabla\mathbf{P})$. This is equivalent to solving a regularized-least squares problem that attempts to fit the data while also minimizing small gradients. We tried this (using conjugate gradient search) but found that the algorithm failed. One interpretation is that the MAP objective function attempts to minimize all gradients (even large ones), whereas we expect natural images to have some large gradients. Consequently, the algorithm yields a two-tone image, since virtually all the gradients are zero. If we reduce the noise variance (thus increasing the weight on the data-fitting term), then the algorithm yields a delta-function for $\mathbf{K}$, which exactly fits the blurred image, but without any deblurring. Additionally, we find the MAP objective function to be very susceptible to poor local minima.

Instead, our approach is to approximate the full posterior distribution $p(\mathbf{K}, \nabla\mathbf{L}_p|\nabla\mathbf{P})$, and then compute the kernel $\mathbf{K}$ with max-

Following Miskin and MacKay [2000], we also treat the noise variance σ^2 as an unknown during the estimation process, thus freeing the user from tuning this parameter. This allows the noise variance to vary during estimation: the data-fitting constraint is loose early in the process, becoming tighter as better, low-noise solutions are found. We place a prior on σ^2, in the form of a Gamma distribution on the inverse variance, having hyper-parameters a, b: $p(\sigma^2|a,b) = \Gamma(\sigma^{-2}|a,b)$. The variational posterior of σ^2 is $q(\sigma^{-2})$, another Gamma distribution.

The variational algorithm minimizes a cost function representing the distance between the approximating distribution and the true posterior, measured as: $KL(q(\mathbf{K}, \nabla\mathbf{L}_p, \sigma^{-2})||p(\mathbf{K}, \nabla\mathbf{L}_p|\nabla\mathbf{P}))$. The independence assumptions in the variational posterior allows the cost function C_{KL} to be factored:

$$<\log \frac{q(\nabla\mathbf{L}_p)}{p(\nabla\mathbf{L}_p)}>_{q(\nabla\mathbf{L}_p)} + <\log \frac{q(\mathbf{K})}{p(\mathbf{K})}>_{q(\mathbf{K})} + <\log \frac{q(\sigma^{-2})}{p(\sigma^2)}>_{q(\sigma^{-2})} \quad (4)$$

where $<\cdot>_{q(\theta)}$ denotes the expectation with respect to $q(\theta)$[2]. For brevity, the dependence on $\nabla\mathbf{P}$ is omitted from this equation.

The cost function is then minimized as follows. The means of the distributions $q(\mathbf{K})$ and $q(\nabla\mathbf{L}_p)$ are set to the initial values of $\mathbf{K}$ and $\nabla\mathbf{L}_p$ and the variance of the distributions set high, reflecting the lack of certainty in the initial estimate. The parameters of the distributions are then updated alternately by coordinate descent; one is updated by marginalizing out over the other whilst incorporating the model priors. Updates are performed by computing closed-form optimal parameter updates, and performing line-search in the direction of these updated values (see Appendix A for details). The updates are repeated until the change in C_{KL} becomes negligible. The mean of the marginal distribution $<\mathbf{K}>_{q(\mathbf{K})}$ is then taken as the final value for $\mathbf{K}$. Our implementation adapts the source code provided online by Miskin and MacKay [2000a].

In the formulation outlined above, we have neglected the possibility of saturated pixels in the image, an awkward non-linearity which violates our model. Since dealing with them explicitly is complicated, we prefer to simply mask out saturated regions of the image during the inference procedure, so that no use is made of them.

For the variational framework, $C = D = 4$ components were used in the priors on $\mathbf{K}$ and $\nabla\mathbf{L}_p$. The parameters of the prior on the latent image gradients π_c, v_c were estimated from a single street scene image, shown in Figure 2, using EM. Since the image statistics vary across scale, each scale level had its own set of prior parameters. This prior was used for all experiments. The parameters for the prior on the blur kernel elements were estimated from a small set of low-noise kernels inferred from real images.

4.1.1 Multi-scale approach

Removing Camera Shake from a Single Photograph
Fergus et al. 2006

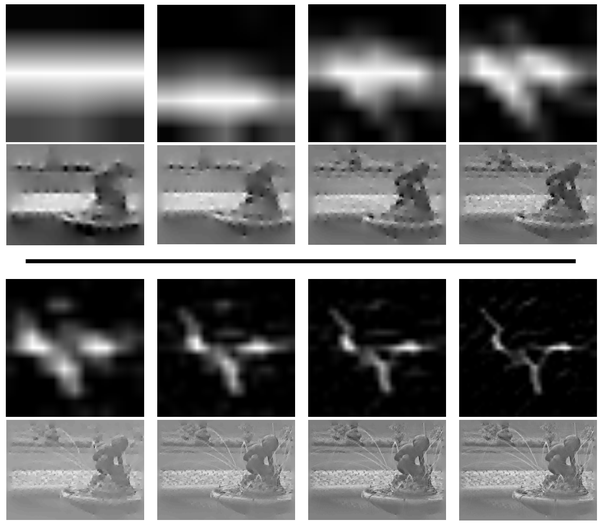


Figure 3: The multi-scale inference scheme operating on the fountain image in Figure 1. *1st & 3rd rows*: The estimated blur kernel at each scale level. *2nd & 4th rows*: Estimated image patch at each scale. The intensity image was reconstructed from the gradients used in the inference using Poisson image reconstruction. The Poisson reconstructions are shown for reference only; the final reconstruction is found using the Richardson-Lucy algorithm with the final estimated blur kernel.

4.1.2 User supervision

Although it would seem more natural to run the multi-scale inference scheme using the full gradient image $\nabla\mathbf{L}$, in practice we found the algorithm performed better if a smaller patch, rich in edge structure, was manually selected. The manual selection allows the user to avoid large areas of saturation or uniformity, which can be disruptive or uninformative to the algorithm. Examples of user-selected patches are shown in Section 5. Additionally, the algorithm runs much faster on a small patch than on the entire image.

An additional parameter is that of the maximum size of the blur kernel. The size of the blur encountered in images varies widely, from a few pixels up to hundreds. Small blurs are hard to resolve if the algorithm is initialized with a very large kernel. Conversely, large blurs will be cropped if too small a kernel is used. Hence, for operation under all conditions, the approximate size of the kernel is a required input from the user. By examining any blur artifact in the image, the size of the kernel is easily deduced.

synthetic test examples, our real images exhibit a range of non-linearities not present in synthetic cases, such as non-Gaussian noise, saturated pixels, residual non-linearities in tonescale and estimation errors in the kernel. Disappointingly, when run on our images, most methods produced unacceptable levels of artifacts.

We also used our variational inference scheme on the gradients of the whole image $\nabla\mathbf{B}$, while holding $\mathbf{K}$ fixed. The intensity image was then formed via Poisson image reconstruction [Weiss 2001]. Aside from being slow, the inability to model the non-linearities mentioned above resulted in reconstructions no better than other approaches.

As $\mathbf{L}$ typically is large, speed considerations make simple methods attractive. Consequently, we reconstruct the latent color image $\mathbf{L}$ with the Richardson-Lucy (RL) algorithm [Richardson 1972; Lucy 1974]. While the RL performed comparably to the other methods evaluated, it has the advantage of taking only a few minutes, even on large images (other, more complex methods, took hours or days). RL is a non-blind deconvolution algorithm that iteratively maximizes the likelihood function of a Poisson statistics image noise model. One benefit of this over more direct methods is that it gives only non-negative output values. We use Matlab's implementation of the algorithm to estimate $\mathbf{L}$, given $\mathbf{K}$, treating each color channel independently. We used 10 RL iterations, although for large blur kernels, more may be needed. Before running RL, we clean up $\mathbf{K}$ by applying a dynamic threshold, based on the maximum intensity value within the kernel, which sets all elements below a certain value to zero, so reducing the kernel noise. The output of RL was then gamma-corrected using $\gamma = 2.2$ and its intensity histogram matched to that of $\mathbf{B}$ (using Matlab's `histeq` function), resulting in $\mathbf{L}$. See pseudo-code in Appendix A for details.

5 Experiments

We performed an experiment to check that blurry images are mainly due to camera translation as opposed to other motions, such as in-plane rotation. To this end, we asked 8 people to photograph a whiteboard[3] which had small black dots placed in each corner whilst using a shutter speed of 1 second. Figure 4 shows dots extracted from a random sampling of images taken by different people. The dots in each corner reveal the blur kernel local to that portion of the image. The blur patterns are very similar, showing that our assumptions of spatially invariant blur with little in plane rotation are valid.

We apply our algorithm to a number of real images with varying degrees of blur and saturation. All the photos came from personal photo collections, with the exception of the fountain and cafe images which were taken with a high-end DSLR using long exposures

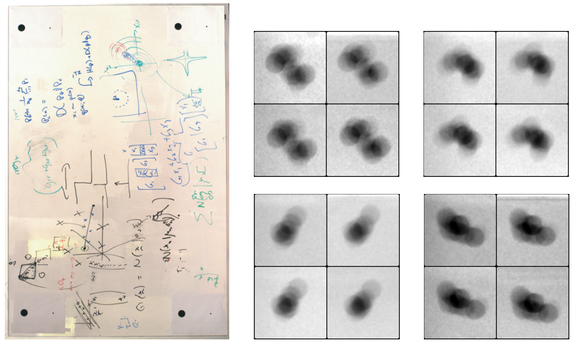


Figure 4: *Left*: The whiteboard test scene with dots in each corner. *Right*: Dots from the corners of images taken by different people. Within each image, the dot trajectories are very similar suggesting that image blur is well modeled as a spatially invariant convolution.

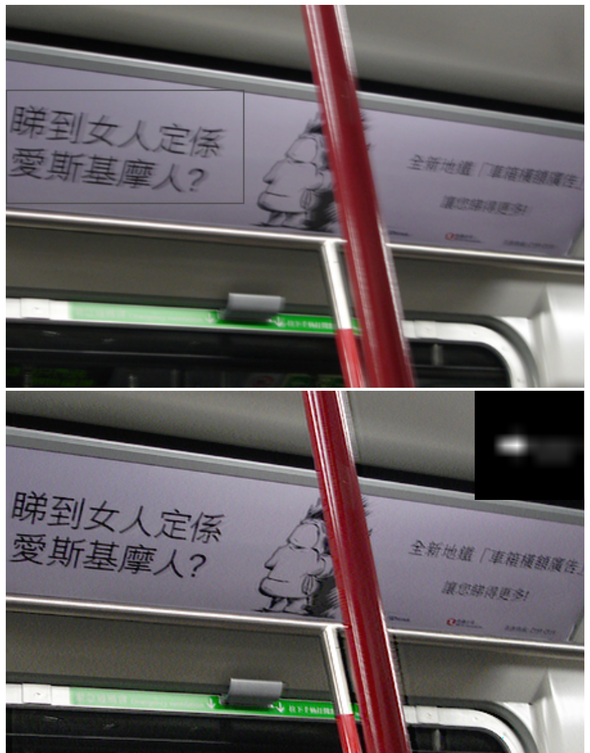


Figure 6: *Top*: A scene with complex motions. While the motion of the camera is small, the child is both translating and, in the case of the arm, rotating. *Bottom*: Output of our algorithm. The face and shirt are sharp but the arm remains blurred, its motion not modeled by our algorithm.

As demonstrated in Figure 8, the true blur kernel is occasionally revealed in the image by the trajectory of a point light source transformed by the blur. This gives us an opportunity to compare the inferred blur kernel with the true one. Figure 10 shows four such image structures, along with the inferred kernels from the respective images.

We also compared our algorithm against existing blind deconvolution algorithms, running Matlab's `deconvblind` routine, which provides implementations of the methods of Biggs and Andrews

Removing Camera Shake from a Single Photograph
Fergus et al. 2006

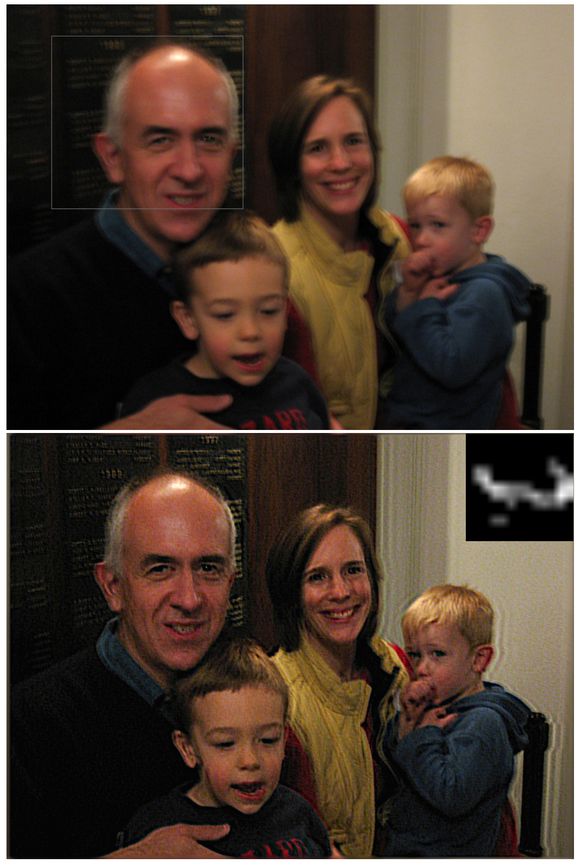


Figure 7: *Top*: A scene with a large blur. *Bottom*: Output of our algorithm. See Figure 8 for a closeup view.

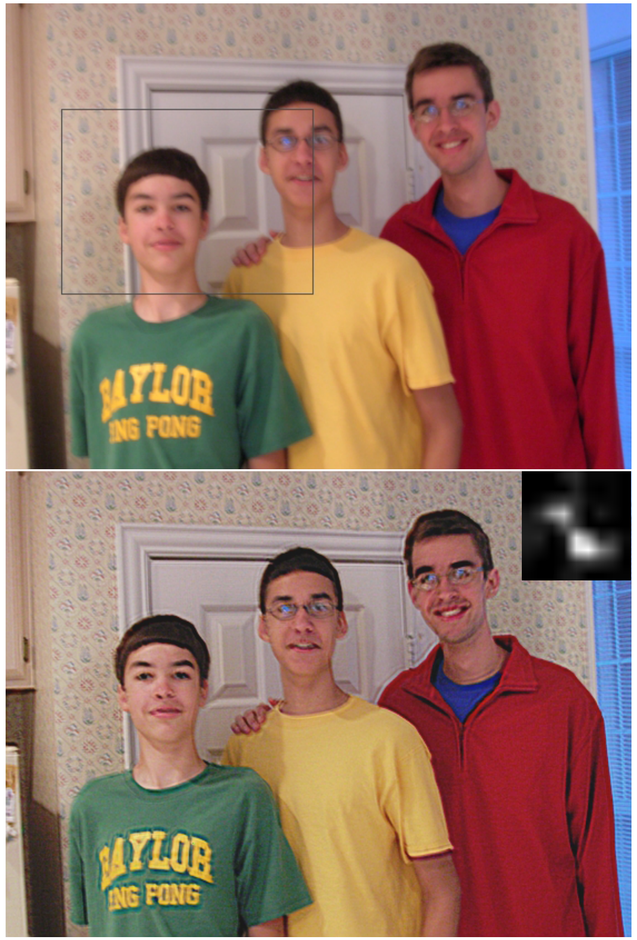


Figure 9: *Top*: A blurry photograph of three brothers. *Bottom*: Output of our algorithm. The fine detail of the wallpaper is now visible.

6 Discussion

We have introduced a method for removing camera shake effects from photographs. This problem appears highly underconstrained at first. However, we have shown that by applying natural image priors and advanced statistical techniques, plausible results can

Figure 10: *Top row*: Inferred blur kernels from four real images (the cafe, fountain and family scenes plus another image not shown). *Bottom row*: Patches extracted from these scenes where the true kernel has been revealed. In the cafe image, two lights give a dual image of the kernel. In the fountain scene, a white square is transformed by the blur kernel. The final two images have specularities transformed by the camera motion, revealing the true kernel.

Figure 11: Baseline experiments, using Matlab's blind deconvolution algorithm `deconvblind` on the fountain image (top) and cafe

Figure 12: *Top*: A blurred scene with significant saturation. The long thin region selected by the user has limited saturation. *Bottom* output of our algorithm. Note the double exposure type blur kernel

Removing Camera Shake from a Single Photograph
Fergus et al. 2006

Acknowledgements

We are indebted to Antonio Torralba, Don Geman and Fredo Durand for their insights and suggestions. We are most grateful to James Miskin and David MacKay, for making their code available online. We would like the thank the following people for supplying us with blurred images for the paper: Omar Khan, Reinhard Klette, Michael Lewicki, Pietro Perona and Elizabeth Van Ruitenbeek. Funding for the project was provided by NSERC, NGA NEGI-1582-04-0004 and the Shell Group.

References

APOSTOLOFF, N., AND FITZGIBBON, A. 2005. Bayesian video matting using learnt image priors. In *Conf. on Computer Vision and Pattern Recognition*, 407–414.

BASCLE, B., BLAKE, A., AND ZISSERMAN, A. 1996. Motion Deblurring and Superresolution from an Image Sequence. In *ECCV (2)*, 573–582.

BEN-EZRA, M., AND NAYAR, S. K. 2004. Motion-Based Motion Deblurring. *IEEE Trans. on Pattern Analysis and Machine Intelligence 26*, 6, 689–698.

BIGGS, D., AND ANDREWS, M. 1997. Acceleration of iterative image restoration algorithms. *Applied Optics 36*, 8, 1766–1775.

CANON INC., 2006. What is optical image stabilizer? `http://www.canon.com/bctv/faq/optis.html`.

CARON, J., NAMAZI, N., AND ROLLINS, C. 2002. Noniterative blind data restoration by use of an extracted filter function. *Applied Optics 41*, 32 (November), 68–84.

FIELD, D. 1994. What is the goal of sensory coding? *Neural Computation 6*, 559–601.

GEMAN, D., AND REYNOLDS, G. 1992. Constrained restoration and the recovery of discontinuities. *IEEE Trans. on Pattern Analysis and Machine Intelligence 14*, 3, 367–383.

GULL, S. 1998. Bayesian inductive inference and maximum entropy. In *Maximum Entropy and Bayesian Methods*, J. Skilling, Ed. Kluwer, 54–71.

JALOBEANU, A., BLANC-FRAUD, L., AND ZERUBIA, J. 2002. Estimation of blur and noise parameters in remote sensing. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing*.

JANSSON, P. A. 1997. *Deconvolution of Images and Spectra*. Academic Press.

JORDAN, M., GHAHRAMANI, Z., JAAKKOLA, T., AND SAUL, L. 1999. An introduction to variational methods for graphical models. In *Machine Learning*, vol. 37, 183–233.

KUNDUR, D., AND HATZINAKOS, D. 1996. Blind image deconvolution. *IEEE Signal Processing Magazine 13*, 3 (May), 43–64.

LEVIN, A., AND WEISS, Y. 2004. User Assisted Separation of Reflections from a Single Image Using a Sparsity Prior. In *ICCV*, vol. 1, 602–613.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2003. Learning How to Inpaint from Global

Appendix A

Here we give pseudo code for the algorithm, `Image Deblur`. This calls the inference routine, `Inference`, adapted from Miskin and MacKay [2000a; 2000]. For brevity, only the key steps are detailed. Matlab notation is used. The Matlab functions `imresize`, `edgetaper` and `deconvlucy` are used with their standard syntax.

Algorithm 1 `Image Deblur`

Require: Blurry image $\mathbf{B}$; selected sub-window $\mathbf{P}$; maximum blur size ϕ; overall blur direction o ($=0$ for horiz., $=1$ for vert.); parameters for prior on $\nabla\mathbf{L}$: $\theta_L = \{\pi_c^s, v_c^s\}$; parameters for prior on $\mathbf{K}$: $\theta_K = \{\pi_d, \lambda_d\}$.

Convert $\mathbf{P}$ to grayscale.

Inverse gamma correct $\mathbf{P}$ (default $\gamma = 2.2$).

$\nabla\mathbf{P}_x = \mathbf{P} \otimes [1,-1]$. *% Compute gradients in x*

$\nabla\mathbf{P}_y = \mathbf{P} \otimes [1,-1]^T$. *% Compute gradients in y*

$\nabla\mathbf{P} = [\nabla\mathbf{P}_x, \nabla\mathbf{P}_y]$. *% Concatenate gradients*

$S = \lceil -2\log_2(3/\phi) \rceil$. *% # of scales, starting with 3×3 kernel*

for $s = 1$ to S **do** *% Loop over scales, starting at coarsest*

  $\nabla\mathbf{P}^s$ = `imresize(`$\nabla\mathbf{P}$`,`$(\frac{1}{\sqrt{2}})^{S-s}$`,'bilinear')`. *% Rescale gradients*

  if (s==1) **then** *% Initial kernel and gradients*

    $\mathbf{K}^s = [0,0,0;1,1,1;0,0,0]/3$. If $(o == 1)$, $\mathbf{K}^s = (\mathbf{K}^s)^T$.

    $[\mathbf{K}^s, \nabla\mathbf{L}_p^s]$ = `Inference(`$\nabla\mathbf{P}^s$,$\mathbf{K}^s$,$\nabla\mathbf{P}^s$,θ_K^s,θ_L^s`)`, keeping $\mathbf{K}^s$ fixed.

  else *% Upsample estimates from previous scale*

    $\nabla\mathbf{L}_p^s$ = `imresize(`$\nabla\mathbf{L}_p^{s-1}$,$\sqrt{2}$,'bilinear')`.

    $\mathbf{K}^s$ = `imresize(`$\mathbf{K}^{s-1}$,$\sqrt{2}$,'bilinear')`.

  end if

  $[\mathbf{K}^s, \nabla\mathbf{L}_p^s]$ = `Inference(`$\nabla\mathbf{P}^s$,$\mathbf{K}^s$,$\nabla\mathbf{L}_p^s$,θ_K^s,θ_L^s`)`. *% Run inference*

end for

Set elements of $\mathbf{K}^S$ that are less than $max(\mathbf{K}^S)/15$ to zero. *% Threshold kernel*

$\mathbf{B}$ = `edgetaper(`$\mathbf{B}$,$\mathbf{K}^S$`)`. *% Reduce edge ringing*

$\mathbf{L}$ = `deconvlucy(`$\mathbf{B}$,$\mathbf{K}^S$,10`)`. *% Run RL for 10 iterations*

Gamma correct $\mathbf{L}$ (default $\gamma = 2.2$).

Histogram match $\mathbf{L}$ to $\mathbf{B}$ using `histeq`.

Output: $\mathbf{L}$, $\mathbf{K}^S$.

Algorithm 2 `Inference` (simplified from Miskin and MacKay [2000])

Require: Observed blurry gradients $\nabla\mathbf{P}$; initial blur kernel $\mathbf{K}$; initial latent gradients $\nabla\mathbf{L}_p$; kernel prior parameters θ_K; latent gradient prior θ_L.

% Initialize $q(\mathbf{K})$, $q(\nabla\mathbf{L}_p)$ and $q(\sigma^{-2})$

Other Writing Tips (I)

- Good resource on mathematical writing:
  “Mathematical Writing”, Donald E. Knuth, Tracy Larrabee and Paul M. Roberts, 1989

- Several specific stylistic (mathematical/prose) elements listed

  - Use consistent notation for the same thing when it appears in several places.

  - Don’t just list a sequence of formulas. Tie concepts together with a running commentary.

  - Don’t be too generous with using notation. Sometimes prose is better.

Other Writing Tips (II)

- Avoid zig-zagging
  - Minimize the number of subject changes. Otherwise, it can be distracting to the reader.
  - Only the central idea(s) should appear multiple times in the paper.
- Bulleted lists are your friend, but don’t go overboard with their use!
- Use examples to illustrate your model/algorithm, especially if it has many moving parts.

Other Writing Tips (III)

- Split long sentences.
- Use footnotes when you want to describe (or excuse) something.
- Central message can appear repeatedly (in paraphrased). Being redundant with the main message is fine.
- Spend time on drawing a good figure especially if you have a pipeline or model with many moving parts.
- Write informative captions for both Tables and Figures.

Case Study: "Attention Is All You Need"

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

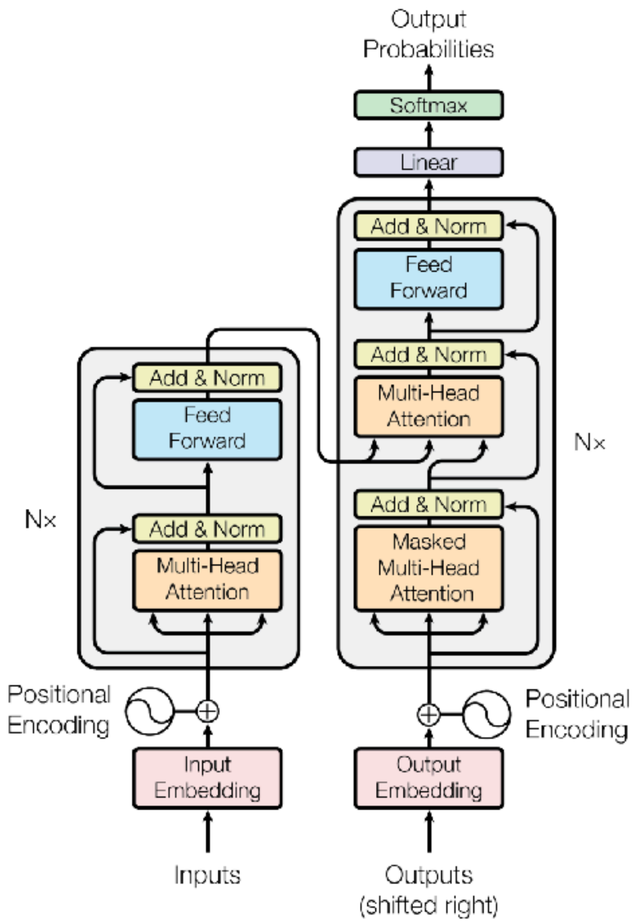


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

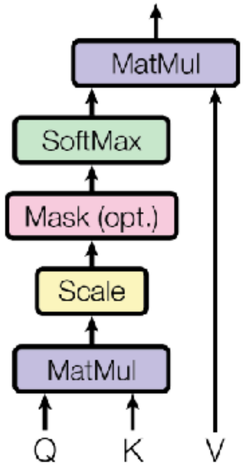
The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

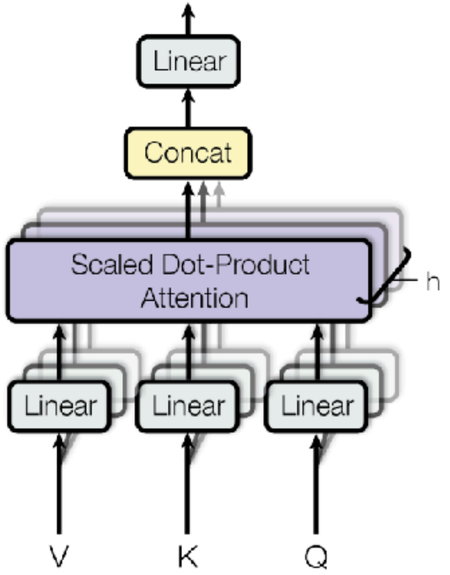
3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional

Scaled Dot-Product Attention



Multi-Head Attention



"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Case Study: "Attention Is All You Need"

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

"Attention is all you need", Vaswani et al., NeurIPS 2017

Useful tools to write a paper in CS

- LaTeX: Powerful document processor. Superior aesthetics in print. Very good at typesetting equations.
  - Good beginner's tutorial: http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf
  - Online, real-time collaborative LaTeX tools like overleaf.com and sharelatex.com are popular
- Compile list of references using a reference management software like BibTeX.

LaTeX tips

- Pay attention to typesetting
  - E.g., make sure mathematical symbols are typeset in mathematical fonts even within text

Let n be an integer

  - Use macros to make your formulas easier to write (and rewrite) consistently

- U
e

```
\newcommand{\G}{\ensuremath{\mathcal{G}}\xspace}
…
Let \G be a class
```

?

Let $\mathcal{G}$ be a class

- Use appropriate packages
- A good online resource: https://en.wikibooks.org/wiki/LaTeX

Reviewing Process

- Anonymous peer review
- In conferences: Reviewing is either blind (only reviewers are anonymous) or double-blind (both authors and reviewers are anonymous)
- 70-80% of papers are rejected at top-tier conferences
  - A poorly written paper almost never makes the cut

Where should I submit my work?

- Important to submit to the right venue. Consult with your advisor.
- You should have read dozens of papers submitted to a venue to understand it better.
  - What does the audience expect?
  - What are the conventions adopted here?
  - What is their process of selection?

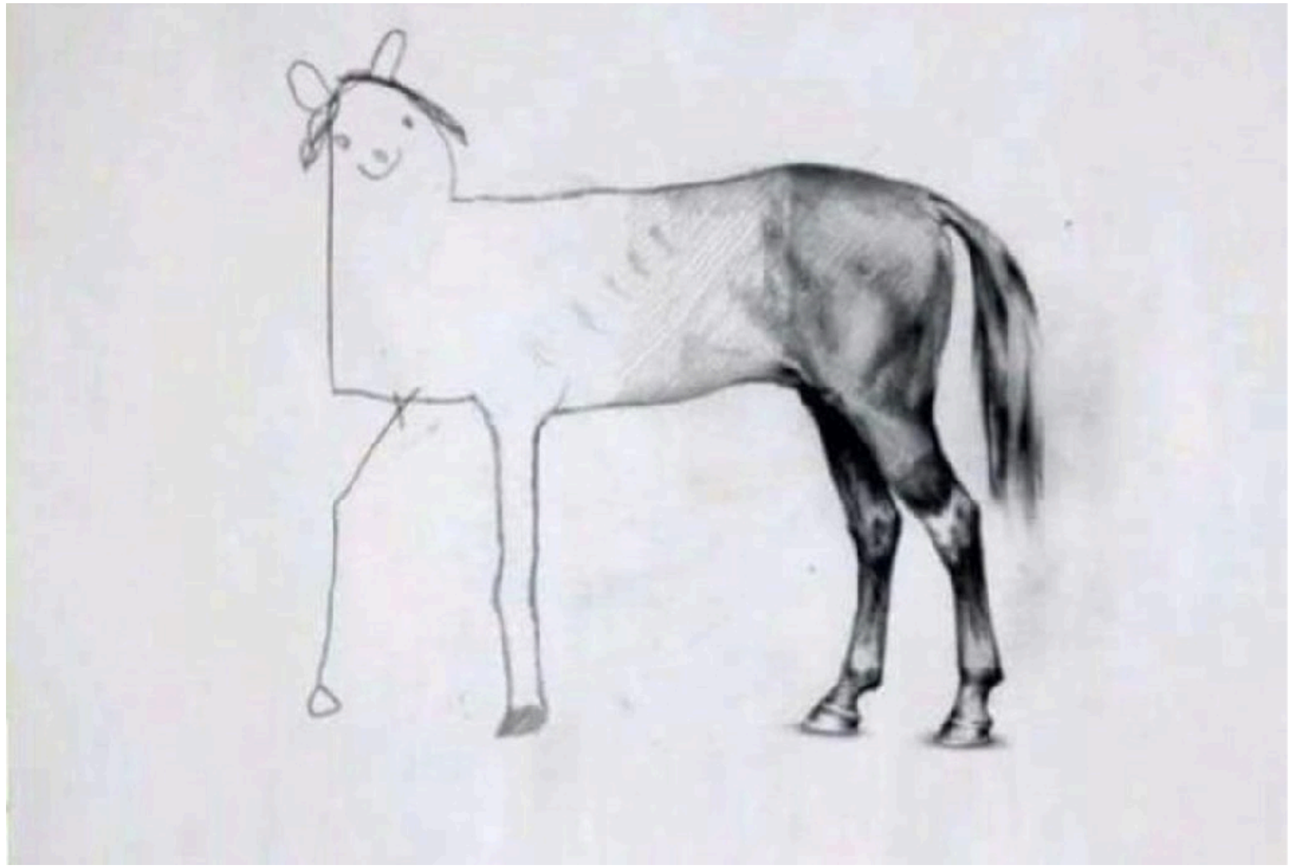
Deciding the author list

- Who are the authors? Everyone who made a significant contribution.
- Author order:
  - Some communities (TCS) use alphabetical order
  - Some communities use descending order of contribution

Rewriting: Lather, Rinse, Repeat

- Writing is an iterative process
  - Don’t get too attached to your words. Rewriting typically produces better text.
  - To allow time for rewrites, start writing your paper early!
- Spend time “debugging” your paper. Use a spell-checker, if needed. Otherwise, very tedious for the reviewer.

Unfortunately, deadlines loom…



- Tip: Spend more time on the important parts of the paper (introduction, overall organisation, key technical aspects)

Final Takeaways

- The reader is the boss. Always keep your reader in mind when structuring your paper.
- Spend time planning your paper. Organisation is key.
  Let good work not sink due to poor writing.
- Rewriting is good writing. Your words are very likely to improve over multiple iterations.
- The old adage “practice makes perfect” holds value when it comes to technical writing.