Layer-Attended Convolutional Networks

Author:
Bird, George `george.bird@student.manchester.ac.uk`

30 November 2022

Abstract

Convolutional networks are highly effective architectures for recognising features in computer-vision tasks. However, they can sometimes obfuscate desirable features if too many convolutional-layers are applied, since their design inclines the transform to not reproduce its input. Thus, Residual Networks were found to be a highly effective adaptation to convolutional networks, which enable the network to preserve an existing optimal arrangement for data-manifolds. In this paper, further issues are described with these existing architectures and a work-in-progress proposal of how to address these problems is outlined. These modifications, for now, remain purely theoretical.

1 Introduction

Convolutional neural networks[1][2] are characterised by a series of layers in which trainable kernals are convolved over a tensor. The purpose is to emphasise relations between nearby elements of a tensor, whilst exhibiting equivariance to where these features occur. As more convolutional layers are applied, there is regularly a gradual shift towards identifying more complicated, yet generalised, abstract classifications of the data[3][4]. Thus convolution results in the (flattened) tensor dimensions changing from representing absolute positions to instead representing positional relations, known as features. Thus incurring a suppression in absolute positional information represented angularly in the activation-space.

This architecture is exceedingly effective at detecting features in real-world images[5], due to an often repeated occurrence of such features throughout the sample and across the larger dataset. Thus, convolutions exploit this redundancy by only utilising a few sets of trainable parameters, which when convolved across the image, are able to detect repeated instances of features. In effect, making the network resilient to the rotation of the embedded data-manifold inside the configuration space. By utilising this spatially equivariant property, the minimised number of tunable-parameters ordinarily experience a faster and more generalised degree of learning. This being a primary factor in the success of convolutions in computer vision. Furthermore, the characteristic repeated features, allowing for successfully shared weights, often persists at increasingly higher complexity scales. Thus, a layered approach to convolutions is typically very effective, with the later layers extracting increasingly abstract features. Figure 1 shows a schematic representation of the convolutional network.

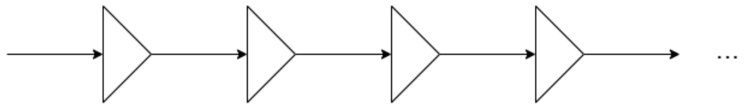

Figure 1: Shows a schematic diagram of a sequential convolutional network. The triangles indicate a convolutional layer.

The intrinsic dimension of the embedded data-manifold must always remain the same (we define as a reformative-transform) or decrease (we define as an abstractive-transform) as it progresses through the convolution layers. As stated, a trade-off between increasingly complex yet generalised classifications and exact positional information is typical. In effect, the expression shift from absolute to relative positions of values of elements. In reformative-transform scenarios, both aspects of the data are preserved in the data-manifold, though typically one is preferentially expressed in a network-interpretable way compared to the other[**?**].

Convolutional network architectures are constructed such that the final layer's information is then propagated to further down-stream networks. As a result the information received by following layers is typically expressed to emphasise the most abstract and complicated features in its representation. It has been shown that sometimes, this most abstract of representations is sub-optimal for network performance - resulting in more convolutional layers impeding function on a task. The Residual Network[6], commonly abbreviated to 'Resnet', is an architecture which combats this problem. It involves an adaptation to the convolution architecture by making a way to bypass a layer. A mixing parameter $\beta \in [0, 1]$ is introduced to perform a weighted average between the initial tensor and the resultant tensor after a convolution is performed on the initial one. This tunable mixing-parameter, can allow the network to easily reproduce the original tensor if the convolutional layer is not necessitated to perform the given task. The resnet weighted average is described in *Eqn.* 1. The success of resnets indicates that lower-layers may produce an optimal representation for a task, where positional information and classification of less abstract-features are more balanced.

$$\vec{g}\left(\vec{x}\right) = \beta \vec{f}\left(\vec{x}\right) \oplus \left(1 - \beta\right) \vec{x} \tag{1}$$

Where $\oplus$ represents an element-wise addition, $\vec{x} \in \mathrm{R}^n$ describes the input tensor (the vector notation needn't imply it must be flattened), $\vec{f} : \mathrm{R}^n \to \mathrm{R}^n$ describes the typical convolutional layer and $\vec{g} : \mathrm{R}^n \to \mathrm{R}^n$ is the function describing residual layer. Figure 2 shows a schematic diagram of the residual network.

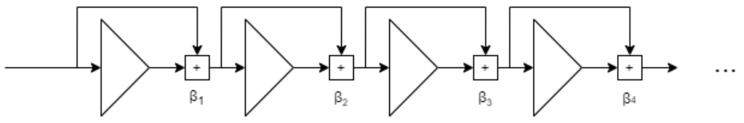

Figure 2: Shows a schematic diagram of a residual network. Where β_i is the mixing-parameter per layer. The upper tracks are weighted by $(1-\beta)$ and the output of the convolutional layer is weighted by β. They are then element-wise summed together, indicated by the square.

In this paper, it is posited that this element-wise addition is still sub-optimal and proposed further modifications are outlined to amend it.

2 Architectural Problems

As mentioned, the element-wise sum in resnets could be problematic. This is because the mixing-parameter can take continuous values and is likely to be in the range $\beta \in (0,1)$ during training. As a result, the combination of two tensors with differing data representation occurs. This could be detrimental to network training. As stated, convolutional layers typically shift the data-manifold into a different representation, which expresses more abstract relations of the original manifold and this is particularly evident when considering its non-linear nature. In resnets, one of the tensors has been passed through an additional convolutional layer with respect to the other. This means that the properties of the manifolds, embedded in each respective tensor, are represented in an inconsistent manner so should not naively be added together. Suggesting that the element-wise average is an inappropriate way to combine these two differing objects.

It is likely that gradient-descent will eventually adapt the transforms, to bring these representations into closer alignment, making the make the weighted average meaningful. Meanwhile, the mixing-parameter is likely to become fixed with a very strong weighting to one tensor as opposed to another resolving the issue. However, in either case, this likely incurs a training slow-down, particularly during early training epochs, and also constrains the network in transforms it can explore during training. Instead we propose that a method involving a concatenation then contraction is a more robust way to bypass the convolutional layer. The contraction will incur another transform, but a layer-attended structure can further alleviate this. In addition, concatenation of different representations of the same initial distribution will not increase the intrinsic dimension of the resultant manifold only the extrinsic dimension of the space, thus contraction can be a lossless transform.

Furthermore, resnets highlighted the potential importance of data-manifolds produced from less abstract convolutional-layers. Though, resnets go someway to making this information accessible to downstream network structures, it is likely limited to the representations produced at a single layer. The sole importance of a single layer's output should not be assumed.

Finally, both convolutions and resnets (at least for $\beta \neq 0$), produce a top-down learning effect in the layers. Not only are the data-manifolds affected by the various transforms, so are the gradients. This problem we define as 'gradient-dilution' where the learning gradients become increasingly averaged out earlier in the sequential convolutional network. In effect, the learning updates applied in the earlier layers are dependent upon the state of the tunable-parameters in the later layers. This likely results in a counterproductive learning-rate limitations in the first few convolutional layers, since the earlier and later convolution layers must train in parallel.

Instead a bottom up approach may be preferable, where effective representations are established in early layers, before being built upon and recombined by parameters in later layers. A switching mechanism, similar to that required for the previously discussed problem, would allow the learning gradients to pass unimpeded to the lowest layers, initiating effective training in these first before progressing to learning increasingly complex representations. Overall, this suggests a layer-attended architecture is preferable, as it allows for the existing training behaviour, if it is indeed optimal, whilst not imposing constraints on this assumption.

3 Architecture of Level-Attended Convolution

The proposed architecture utilises the advantages of the convolutional layers and identity-transforms encouraged by resnets, whilst also allowing the adaptive readout of information from any layer. In effect, both standard convolutional networks and resnets can be viewed as a special case of the following generalised architecture.

The level-attended convolutional network, outlined in *Fig.* 3, also consists of sequential convolutional layers as found in standard convolutional networks. However, after each layer the tensor is duplicated, with one tensor fed into the next layer and the other being outputted. For n convolutional layers, this produces a set of n, potentially variously sized, outputted tensors.

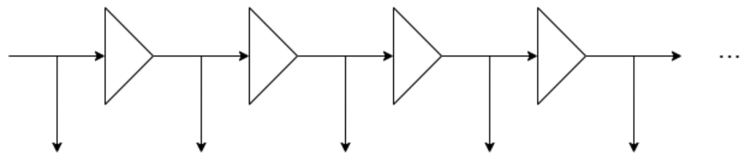

Figure 3: Shows a schematic diagram of a convolutional network where the resultant tensor from each convolutional layer is duplicated and outputted.

At this stage a multitude of possible attention mechanism and further transforms can be applied the outputs. The various combinations should be trialled for effectiveness. In addition, each layer will have a data-manifold with a unique representation; there is likely a conflict between the optimal representation for each of the duplicated tensors. Therefore, it is probably sensible to pass the outputted tensor through a fully-connected network so that this conflict is resolved and a different representation for each tensor is possible. This transform would ideally be reformative, to not vastly alter how abstractly the data is represented. This is fully-connected transform is outlined in $Fig.$ 4

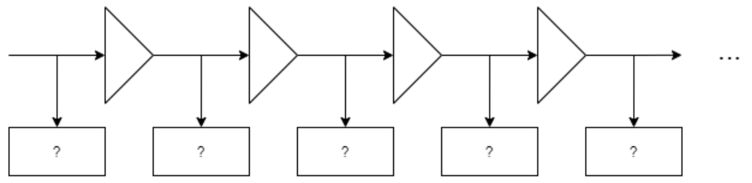

Figure 4: Shows a schematic diagram of a convolutional network where the resultant tensor from each convolutional layer is duplicated and outputted then passed through a fully-connected network.

Further to this, a concatenation and contraction, where each tensor is appropriately weighted in the resultant output, should be implemented. This weighting, is a form of layer-wise attention and can be static or dynamic. This enables a bottom-up approach to learning, where the simplest layers can trained first and a progressive shift in learning towards higher abstractive layers can occur if this is preferable for the network. Thus, the network will appropriately train to the desired level of abstraction without having to train all layers first before this realisation is made. A few examples of various attention-mechanisms are given below, though this list is not exhaustive and many possibilities should be trialled to find the most effective.

- "Innate-Static": this consists of a concatenation of all the output tensors, into a larger tensor. This larger tensor could then be passed through a many-to-few fully-connected network. In this case the fully-connected network is tuned through training to select the most important data from each layer. Data from any and all layers can be represented in the final output. In this scenario, there is no necessity for the aforementioned fully-connected network before concatenation. This is outlined in $Fig.$ 5.

- ”Static Trainable-Vector”: this form more closely aligns with the resnet model. A vector of tunable-parameters, with the properties $\vec{x}_i \in [0, 1]$ and $\sum_i \vec{x}_i = 1$, could be split such that each element weights each output tensor before concatenation and contraction.
- ”Self-Attention”: Similar to those found in transformer networks[7], an attention mechanism could be applied to dynamically select which aspects of the data are relevant for the current forward pass of the network. Thus, the network can shift attention to whichever layer is currently needed.
- ”Dynamic-Recurrent”: As discussed in the recurrency-abstraction theorem[8], recurrent neural networks always incur a form of attention when processing its inputs. This dynamic attention could be implemented by placing a recurrent neural network, such as an long-short term memory network[9], after the concatenation. This can be utilised by the network to select which information can propagate further through the network.

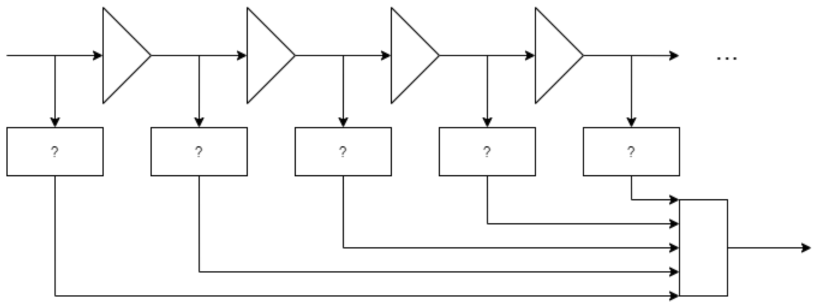

Figure 5: Shows a schematic diagram of a convolutional network where the resultant tensor from each convolutional layer is duplicated and outputted. The outputs are then concatenated and then passed through a fully-connected network.

Figure 6 shows a schematic diagram of a general attention mechanism described in the latter three described attention mechanisms.

It can be seen that all above mechanisms allow bottom-up or top-down training for the network, depending on which is most advantageous. Moreover, the network can access whichever layer’s representation is most fruitful for the task.

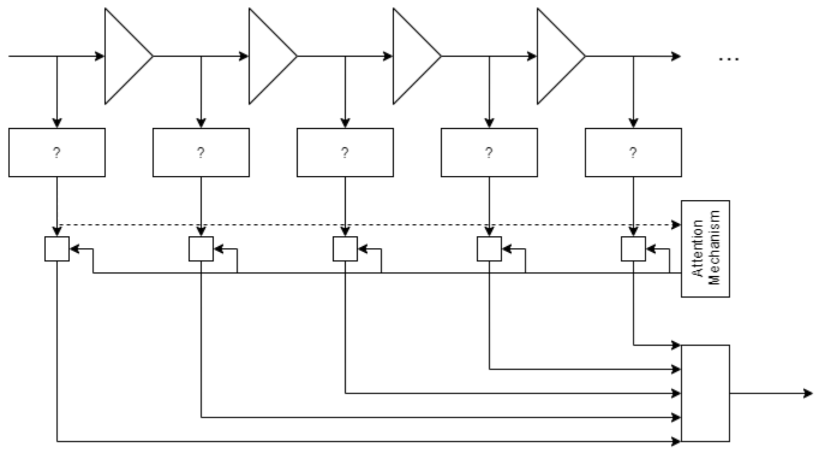

Figure 6: Shows a schematic diagram of a convolutional network where the resultant tensor from each convolutional layer is duplicated and outputted. The outputs are then passed through an attention mechanism, concatenated and then contracted by some means. The dashed line shows dependencies for the case of dynamic attention.

Furthermore, the concatenation alleviates the issue of mismatching representations in the resnet's element-wise summation.

Since both resnets and standard convolutional networks are special-cases of this architecture, this architecture can be viewed as a reduction in assumptions. The aim is that this structure continues to use the benefits of convolutional layers to reduce redundancies, whilst enabling access to any layer's representation, like resnets, as well as also enabling more adaptive layer training and preventing addition of mismatching data-manifolds found in resnets. Furthermore, the accessibility of each layer's information means that downstream corruption of the data by a anomalous training or unfortunate initialisation does not impede the functionality and training of the network overall.

4 Expected Behaviour

It is expected that a a shift in layer-functionality is observable throughout training by monitoring the degree of attention per layer. We would expect that after initialised, with equal attention weighting, an initial shift towards prioritising information from the lower-most level would be observed. Hence reducing the effect of gradient dilution and training dependency on higher layered kernals. Once the lower most layer is becoming sufficiently trained and yielding meaningful manifolds, we would expect to see a shift in attention to the next layer if increased abstraction is necessitated. It is likely that attention is present on all layers during training, but latter convolutional layers being suppressed to have negligible contribution earlier in the training.

Thus throughout training the attention shifts to increasingly higher layers until sufficient abstraction is performed on the manifold for the network to optimally achieve its task. At this time, or before, the mixing-parameters will redistribute to their preferred arrangement.

Standard convolution networks are inspired by the human visual system, likewise the proposed layer-attentive network may be further improved by comparing relations to visuospatial neglect and hemianopia in humans. This will be further explored in future revisions to this work, in addition to further testing and refinement of the new proposed network.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[3] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *CoRR*, vol. abs/1506.06579, 2015.

[4] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Technical Report, Univeristé de Montréal*, 01 2009.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[8] G. Bird and M. E. Polivoda, "Backpropagation through time for networks with long-term dependencies," 2021.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.