Why Do Convolutional Neural Networks Design Better Buildings?

Aleksandra Jastrzębska*, Stanisław Jastrzębski**

Extended Abstract

Deep neural networks (DNNs) have recently shown promise for applications such as weather forecasting Agrawal et al. [2019] or discovery of new materials. DNNs can find better solutions in shorter times, compared to previous approaches. For example, Agrawal et al. [2019] demonstrate that a DNN can predict precipitation two orders of magnitude faster than a state-of-the-art physics-based approach.

This trend is present in structural optimization as well. Structural optimization is an important technique for designing objects such as bridges or skyscrapers. A common approach is to represent a given design (e.g. a bridge) as a density on a grid. Next, an off-the-shelf optimizer can be used to find optimal (according to a user-specified objective) design.

The above procedure relies on how the optimization problem is *parametrized*. The simplest approach is to represent each point (pixel) on the grid as a separate variable. Instead, Hoyer et al. [2019] have proposed to represent the density using a convolutional neural network (CNN). The CNN output is treated as the design. Counter-intuitively, Hoyer et al. [2019] have shown that using CNN as a reparametrization enables finding better solutions compared to optimizing directly in the pixel space.

The authors proposed that the reason for this is that CNNs bias the solution towards *simpler* design that exploits larger scale structures. In this work, we delve deeper into this direction. Our contributions are as follows:

- We show that using the CNN reparametrization for the initial part of training (first 8 steps out of 200), and then switching to pixel-based optimization, maintains most of the benefit of CNN reparametrization.
- Furthermore, we show that switching to pixel-based optimization towards the end of training enables finding an even better solution (40% relative error reduction).

Experiments

Our experimental setup largely follows Hoyer et al. [2019]. We focus on two models from Hoyer et al. [2019]: CNN-LBFGS, and Pixel-LBFGS. Both models are optimized using the L-BFGS optimizer. We select 52 random tasks from Hoyer et al. [2019]. For computational reasons, we removed tasks that have more than 200 pixels. All models are optimized for 200 steps. We will refer to the two models as CNN and Pixel, respectively.

We examine the effect of switching from CNN to Pixel. We take L ($L \in \{1, 2, 4, 8, 32, 64\}$ steps using CNN, and then take the remaining $200 - L$ steps using Pixel. We will refer to these models as Switch-L. For each model, we report the mean normalized performance across tasks. Following Hoyer et al. [2019], we normalize the performance on each task to $[0, 1]$ range.

Table 1 reports the results. We make two observations. First, for $L = 8$ we already close 80% of the gap between CNN, and Pixel. This is remarkable, given that 8 steps consititute only 6% of the overall number of steps (200). Second, using $L = 64$ further improves upon CNN. This taken together suggest that the main role of CNN reparametrization is to bias the optimization in the early phase of training. In the late of training, it even seems to hurt the quality of the solution.

*A. B. Engineering PC. Writing, implementation, running experiments.
**New York University. Conceptualization, writing.

CNN	Pixel	Switch-1	Switch-2	Switch-4	Switch-8	Switch-16	Switch-32	Switch-64
0.00011	0.00142	0.00073	0.00073	0.00068	0.00037	0.00014	0.0001	**0.00007**

Table 1: The mean normalized score (the lower the better) for each method, averaged over 52 tasks.

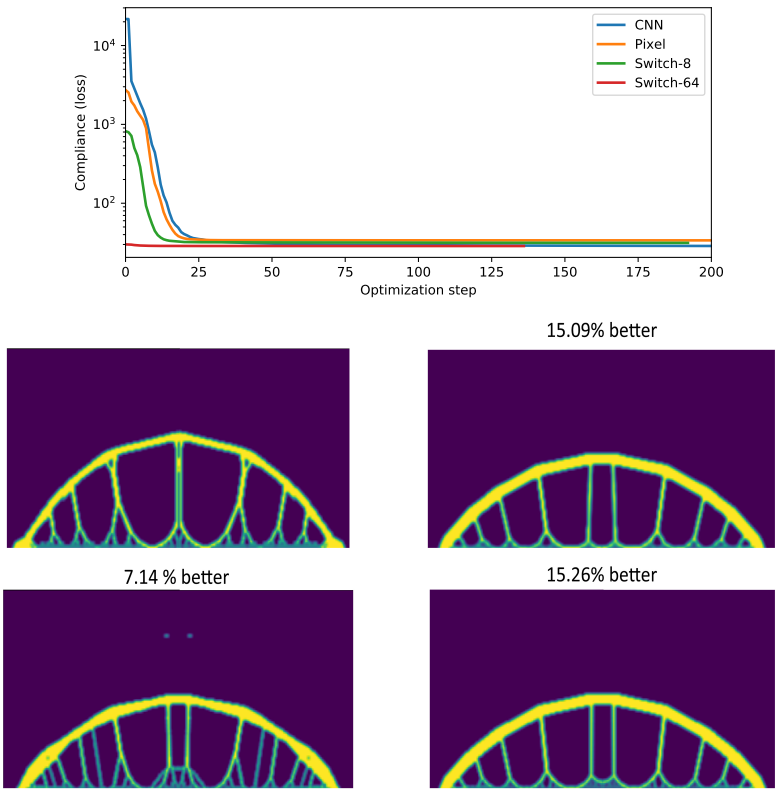

Figure 1: Results on a selected task ("anchored suspended bridge 128x128 0.1"). Top: loss during training. Bottom: designs found (from top left to bottom right) by Pixel, CNN, Switch-8 and Switch-64, respectively.

In Figure 1, we show training curves from one of the tasks. On this task, Switch-64 achieves lower (better) final score. Additionally, in agreement with Hoyer et al. [2019], the solution found by Pixel is visibly more complex than CNN, in the sense that it finds more intricate geometry.

Conclusions In this work, we have shown that CNN reparametrization matters more in the early phase of training, then in the late phase. In particular, switching to Pixel-LBFGS after around 30% steps, enables finding even better solution.

This opens avenue for future work for designing faster optimizers that rely on CNN-based reparametrization only in the early phase of training. Alternatively, our gained understanding might help designing explicit regularizers that do not rely on deep neural networks.

References

Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. Machine learning for precipitation nowcasting from radar images, 2019.

Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. Neural reparameterization improves structural optimization, 2019.