Towards Robust Image Classification Using Sequential Attention Models - Supplementary Material

Anonymous CVPR submission

Paper ID 7672

1. Model details

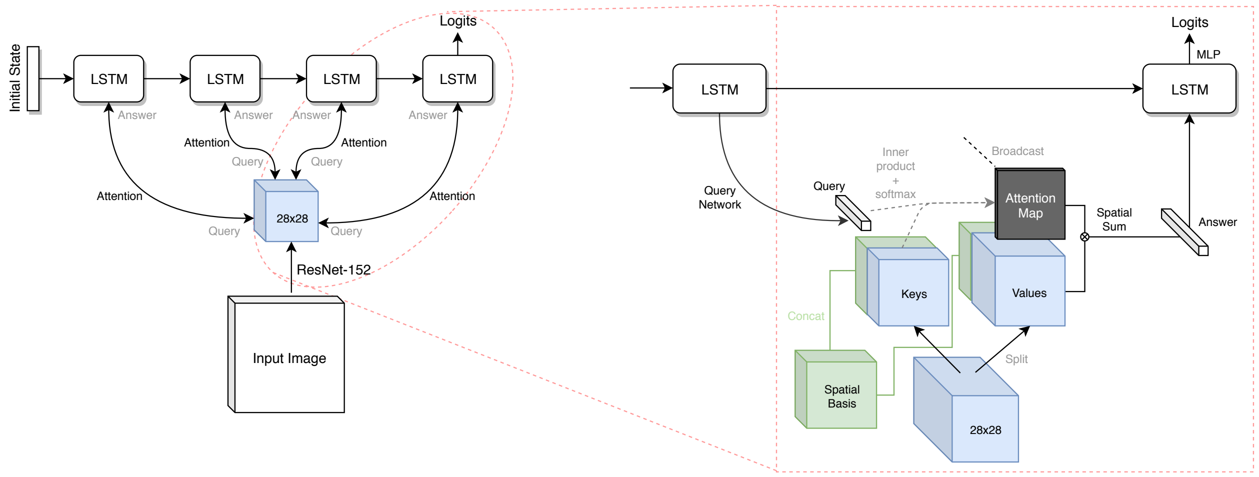

Figure 1: Full model details. Left - high level overview of the recurrent mechanism (for a 4-step model) and the corresponding vision net. Right - a zoomed in detail on one attention step. See text for full details.

1.1. Detailed model overview

Here we detail and expand on the specifics of the model described in the main paper. For a more thorough overview and discussion, we refer the reader to [2].

Figure 1 depicts the model in full detail. On the left is the unrolled compute graph for a 4 step model. On the right is a detailed overview of the attention mechanism for a single attention head.

The model is comprised of two main components - a “vision net” component which processes the input image and outputs a spatial tensor (of size 28×28 in this case) and a “controller” component, which is a recurrent core attending this tensor at every time step. On the left of Figure 1 we can see the image being processed through the vision net (ResNet-152, in this case) and the output tensor being produced. This happens only once per image since the input image is fixed for all compute steps. At the top we see the controller (in this case an LSTM) unrolled. A learned initial state is used for the first time step. At each time step the model attends the vision nets’ output tensor, sending out a query vector(s) and receiving an answer vector(s) to process. At the final step we read out the controller’s state, pass it through an MLP and produce the class logits.

The right side of Figure 1 depicts the inner workings of the attention mechanism. The output tensor from the vision net is split into two tensors along the channel dimension. The first C_k channels ($C_k = 32$ in all our experiments) are used as the “keys tensor” (of size $28 \times 28 \times C_k$). The rest of the channels are used as the “values” tensor with C_v channels ($C_v = 2016$ in all experiments, tensor size is $28 \times 28 \times C_v$). To both the keys and values tensors we concatenate a “spatial basis” tensor of size $28 \times 28 \times C_s$. This fixed tensor allows the attention to preserve and query about spatial information after the spatial structure is lost (see below).

We produce a "query" vector by passing the previous controller state (together with the last time step's logits) through a "query network" (an MLP). This query vector is of size $1 \times 1 \times (C_k + C_s)$. We take the inner product of the query vector with each *spatial* position in the keys tensor to produce a 28×28 single channel attention logits map. We pass this logits map through a spatial softmax to produce the final attention map for this head.

We broadcast the attention map along the channel dimension and point-wise multiply it with the values tensor. The result is then reduce summed along both spatial dimensions to produce the "answer" vector of size $1 \times 1 \times (C_v + C_s)$. The importance of the spatial basis becomes clear at this point, since the summation causes all spatial structure to be lost.

We concatenate all answer vectors together with all query vectors and use these as inputs to the controller at the current time step. We pass the controller state through an MLP to produce the class logits. If it is the last time step, we produce the final class logits and the loss is calculated from them.

1.2. Model parameters

We now give the full details of model parameters. Unless otherwise noted all non-linear activations are ReLUs. All learned parameters are initialized randomly.

1.2.1 Vision net

For the vision net we use a standard ResNet-152 V2. We change the stride to 1 in all residual blocks other than the second one and we don't use the final linear layer. The output of this network for a 224×224 image is a $28 \times 28 \times 2048$ tensor.

1.2.2 Controller

The controller is a standard LSTM with a hidden size of 1024 units. The initial state is randomly initialized and learned with the rest of the network.

1.2.3 Query network

The query network is a standard MLP with one hidden layer of size 1024. Its input is the same size of the controller state plus the size of logits (2024 in our case). Its output size is $(C_k + C_s) \times N$ where N=4 is the number of heads we use. The spatial basis has 64 channels, hence the total output size is 384 (with 32 key channels).

1.2.4 Output MLP

This is a single hidden layer MLP with 1024 units in the hidden layer. Its input is the hidden state of the controller and its output is 1000 units large (i.e. number of classes).

1.2.5 Spatial Basis

The spatial basis is a fixed tensor of size $28 \times 28 \times C_s$. We use spatial sine and cosine functions to form the basis, with 4 frequencies per dimension resulting in $C_s = (2 \times 4)^2 = 64$ channels.

2. Training and evaluation details

We follow the training and evaluation protocol of [3] in most parameters. We use a gradient descent optimizer with momentum. We use a lower learning rate of $0.5e^{-1}$ scaled by a batch size of 256. For our models we use batch size of 1024 so the initial learning rate is 0.2. We warm start for the first 5 epochs and then anneal the learning rate (with a factor of $1e^{-1}$) after 35 epochs, 70 epochs and 95 epochs. We train for 120 epochs.

The attack during training is random targeted PGD with signed gradients. We use $\epsilon = 16$, random initialization probability of 0.8 and step size of 1/255. The loss function for adversarial training is only on adversarial examples, there is no loss coming from clean images. We add L_2 weight decay loss with weight $1e-4$, and label smoothing of 0.1. We use the same attack in evaluation, with step size of $1/255$ for all number of steps other than 10 where we use $1.6/255$, again following [3].

Model	5000 steps		10,000 steps	
	Top-1 Accuracy	Attack Success Rate	Top-1 Accuracy	Attack Success Rate
ResNet-152	28.12%	58.03%	27.64%	58.79%
S3TA-2	32.85%	52.50%	32.63%	52.94%
S3TA-4	35.67%	50.57%	35.43%	50.96%
S3TA-8	36.20%	50.75%	35.44%	51.91%
S3TA-16	37.36%	49.50%	36.96%	50.23%
ResNet-152-30	36.69%	41.33%	36.78%	41.25%
DENOISE	42.76%	33.83%	42.99%	33.88%
S3TA-16-30	**46.11%**	**24.91%**	**46.20%**	**24.68%**

Table 1: Results for extra strong PGD attacks with 5000 and 10000 steps. The general picture remains as S3TA-16-30 is significanly more robust than all other models. S3TA-16 trained with 10 PGD steps is more accurate than a ResNet-152 trained with 30 PGD steps even under these powerful attacks.

Attack steps	100 steps			1000 steps		
Random restarts	1	10	100	1	10	100
ResNet-152-30	39.77%	39.37%	39.15%	37.41%	36.51%	35.75%
DENOISE	46.16%	44.68%	44.68%	43.57%	41.91%	41.37%
S3TA-16-30	**47.76%**	**47.27%**	**47.09%**	**46.74%**	**46.09%**	**45.78%**

Table 2: PGD attack with multiple restarts. We test the various method under attacks starting at multiple random initialization points. We test with 100 and 1000 PGD attack steps, initialized at 1, 10 and 100 random starting points. We count the attack successful even if only one of the adversarial examples caused the model to output the target class. We only count an image to be correctly classified if it was correctly classified across all adversarial images. As can be see, all models are quite robust here, though again, S3TA-16-30 outperforms them all.

3. Additional results

In Table 1 we report results on extra strong attacks with 5000 and 10,000 steps — due to the cost of evaluation we evaluated only on 2200 random images from the ImageNet test set, but in our experience numbers change very little on larger evaluation sets.

3.1. PGD attacks with multiple restarts

Here we test the three strongest models against attacks with multiple restarts. We test with the same PGD with signed gradient attack used so far (with a 100 and 1000 steps) but we allow the attacker to start from multiple random initialization: 1, 10 or 100. We take the strongest example from all restarts to measure the top-1 and attack success rate (i.e. even if one of the adversarial attacks succeeds we count it as a success, and if the network mislabels a single image out of the attacks we do not increase the top-1 score). Results are reported in Table 2. As can be seen, all models are quite robust to multiple restarts, and S3TA-16-30 is better than the other models.

3.2. Random targeted attacks with an Adam optimizer

It has been observed [1] that random targeted attacks using the FGSM method are sometimes weaker than attacks optimized with the Adam optimizer. Here we test the performance of the model with this potentially stronger type of targeted attack. We use a 250 step attack, optimized with the Adam optimizer with learning rate annealing (0.1 until step 100, 0.01 until 200 and 0.001 for the last 50). This has been shown [1] to be quite a strong attack. Results are reported in Table 3. As can be seen all models perform worse here than with PGD with iterative FGSM, however S3TA-16-30 is significantly more robust here as well.

	Top-1	Success rate
ResNet-152-30	26.38%	42.94%
DENOISE	32.99%	35.04%
S3TA-16-30	**36.83%**	**27.00%**

Table 3: Random targeted PGD attack with Adam optimizer. We measure the robustness of some models under an random targeted attack optimized with the Adam optimizer (rather than iterative FGSM). The attack uses 250 iterations, with learning rate 0.1 for the first 100 iterations, 0.01 for next 100 and 0.001 for the last 50. This has be shown [] to be a stronger attack the iterative FGSM. As can S3TA-16-30 is significantly better at defending here compared to the other models.

4. More adversarial examples

4.1. Non visible examples

Figure 2 shows adversarial examples for a S3TA-4 model where there is no visible structure to be seen, as is often the case with fully trained models.

4.2. Structured examples - mid training

Figure 3 shows adversarial examples for a S3TA-16 model around half-way through training, with clear salient global structures. These are much more common before training concludes (but still appear in fully trained models as in the main paper). Understanding the exact circumstances under which these appear is an open question.

References

[1] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing, 2019. 3

[2] Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J Rezende. Towards interpretable reinforcement learning using attention augmented agents. *arXiv preprint arXiv:1906.02500*, 2019. 1

[3] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *CoRR*, abs/1812.03411, 2018. 2

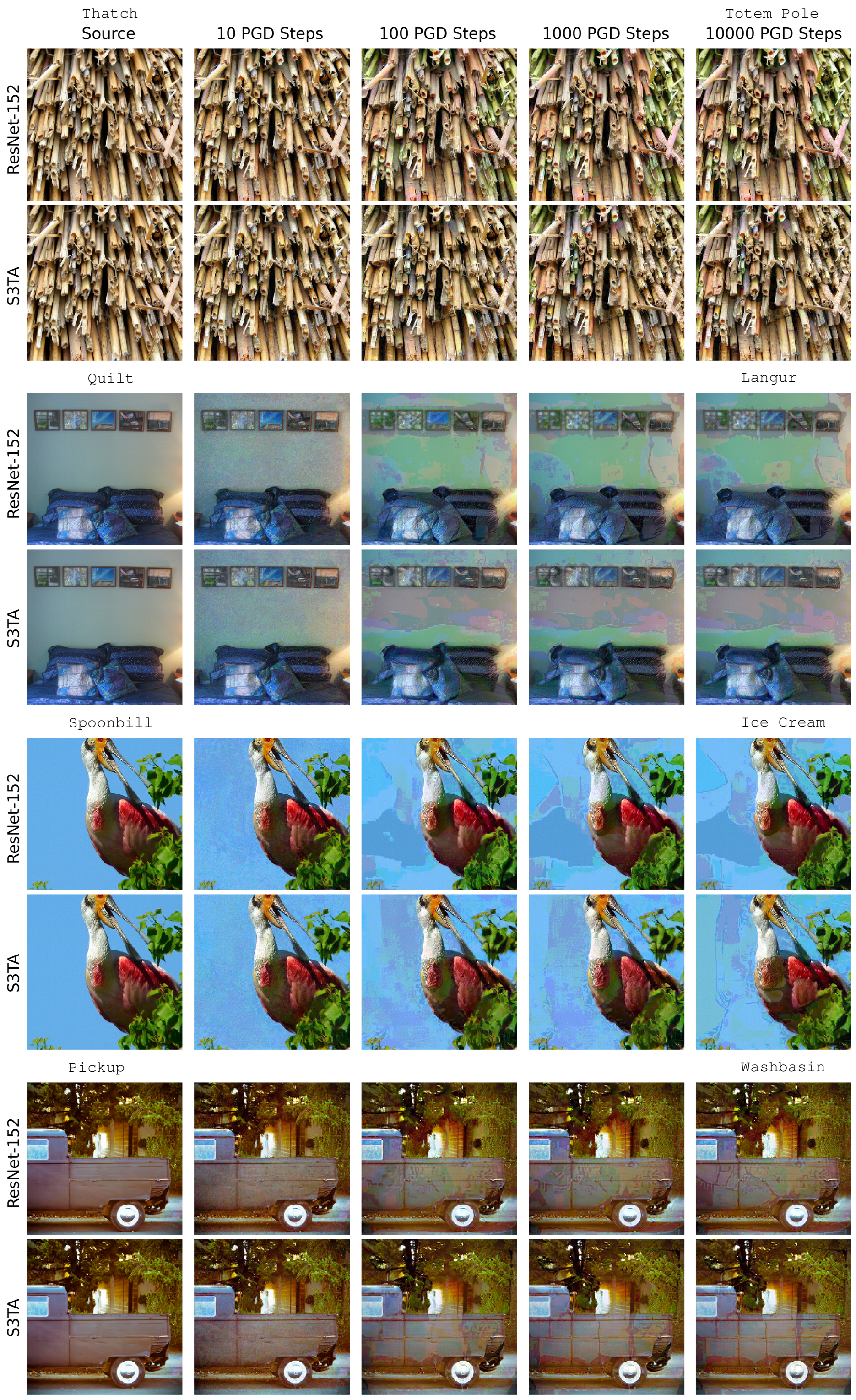

Figure 2: Adversarial images generated by PGD to attack the ResNet-152 and S3TA-4 model with no apparent visual structure. The source label is depicted on the left, the target label on the right. Source images are the leftmost column. We generate examples for 10, 100, 1000 and 10,000 PGD steps. These are cases where the perturbation does not have clear structure, as for many of the images. Some local structure may be interpretable - tiles for the washbasin for example, but there's little coherent global structure. Images are best viewed on screen and zoomed in.

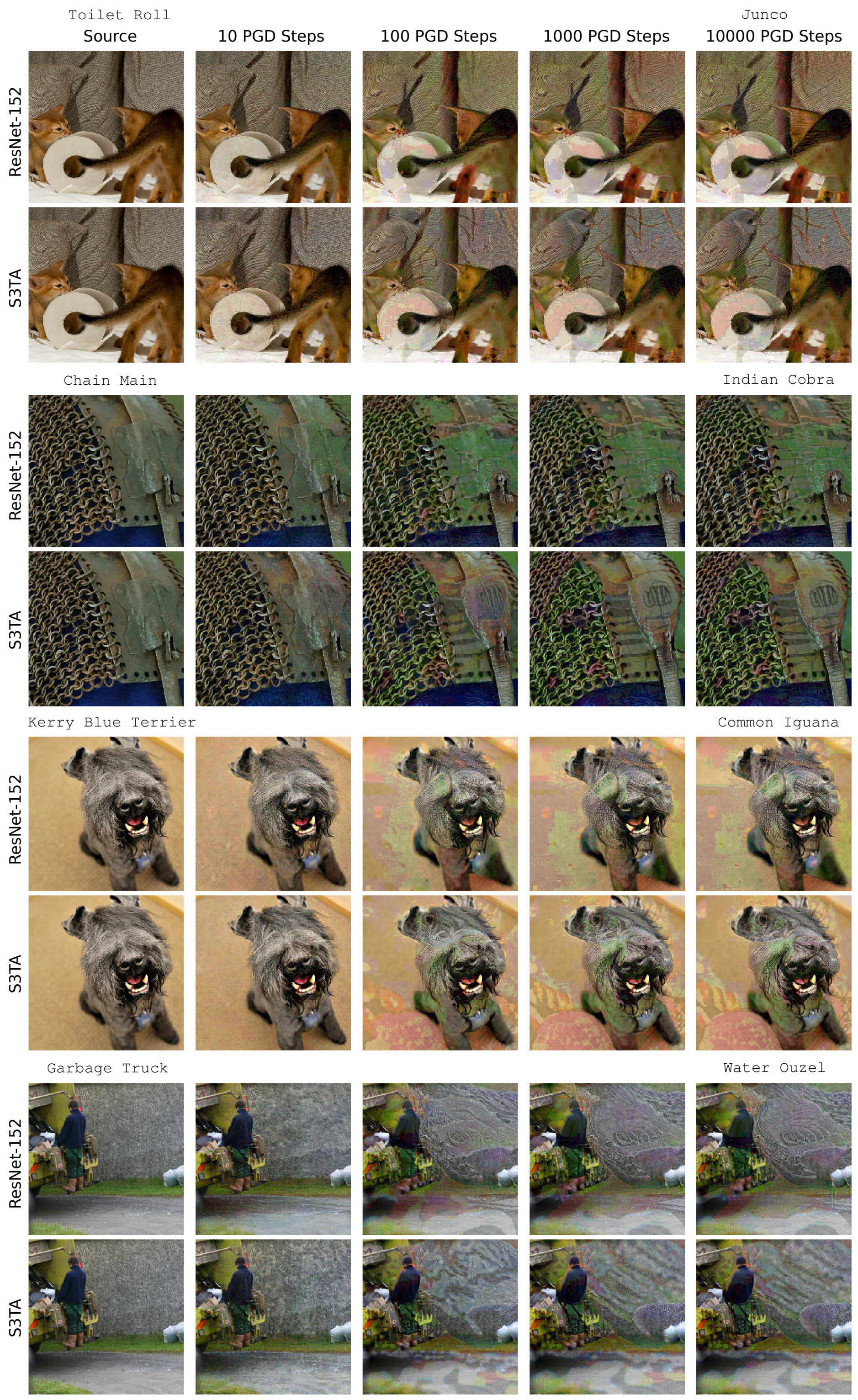

Figure 3: Adversarial images generated by PGD to attack the ResNet-152 and S3TA-4 model with clear and intrepretable structure. The source label is depicted on the left, the target label on the right. Source images are the leftmost column. We generate examples for 10, 100, 1000 and 10,000 PGD steps. While the ResNet attacks are not particularly interesting, the S3TA examples are extremely structured. Note the salient, global and often realistic structures appearing - the Junco bird, the head of the Iguana, the body of the Indian Cobra or the Water Ouzel, a bird which is often depicted next to water (which is also created). We encourage the reader to view these on screen and zoomed-in.