On the Functional Similarity of Robust and Non-Robust Neural Representations

András Balogh [1] **Márk Jelasity** [1 2]

Abstract

Model stitching—where the internal representations of two neural networks are aligned linearly—helped demonstrate that the representations of different neural networks for the same task are surprisingly similar in a functional sense. At the same time, the representations of adversarially robust networks are considered to be different from non-robust representations. For example, robust image classifiers are invertible, while non-robust networks are not. Here, we investigate the functional similarity of robust and non-robust representations for image classification with the help of model stitching. We find that robust and non-robust networks indeed have different representations. However, these representations are compatible regarding accuracy. From the point of view of robust accuracy, compatibility decreases quickly after the first few layers but the representations become compatible again in the last layers, in the sense that the properties of the front model can be recovered. Moreover, this is true even in the case of cross-task stitching. Our results suggest that stitching in the initial, preprocessing layers and the final, abstract layers test different kinds of compatibilities. In particular, the final layers are easy to match, because their representations depend mostly on the same abstract task specification, in our case, the classification of the input into n classes.

1. Introduction

Methods for studying the similarity of the internal representations of different neural networks are important tools for understanding representations. Several such methods are based on statistical approaches like Centered Kernel Alignment (CKA) (Kornblith et al., 2019) or the two variants of Canonical Correlation Analysis (CCA) (Raghu et al., 2017; Morcos et al., 2018).

More recently, a new class of methods based on *model stitching* have emerged that concentrate on *functional* similarity (Lenc & Vedaldi, 2019; Csiszárik et al., 2021; Bansal et al., 2021). This methodology focuses on the question whether the representation of one network can substitute the representation of another network with the help of a simple (often linear or affine) transformation, while keeping some functional property such as classification accuracy.

In these works, it has been argued that functional similarity does not necessarily correlate with the statistical measures of similarity and, more importantly, the internal representations seem to be *surprisingly compatible* in a large number of different scenarios in this functional sense.

At the same time, there is a general consensus that adversarially robust networks (Szegedy et al., 2014; Goodfellow et al., 2015) do have functionally different representations. For example, robust networks are invertible as opposed to non-robust networks (Engstrom et al., 2019b), their generalization seems to rely on an entirely different feature set (Ilyas et al., 2019), and they cluster their representations differently (Bai et al., 2021).

Model stitching is a promising empirical tool to shed more light on the differences between robust and non-robust representations. It allows us to gain insight into the function of each layer of a network architecture, and we can study not only accuracy, but other functional properties as well, such as robust accuracy. Motivated by this, we adopt the model stitching approach to investigate a specific question: *do adversarially robust networks have functionally similar representations to those of non-robust networks?*

1.1. Our Contributions

In order to answer our research question, we performed an empirical study of stitching adversarially robust models and non-robust models in a number of different scenarios, for the image classification task[1]. Here, we list our most important findings and contributions.

[1]University of Szeged, Hungary [2]ELKH SZTE Research Group on Artificial Intelligence, Szeged, Hungary. Correspondence to: András Balogh <abalogh@inf.u-szeged.hu>, Márk Jelasity <jelasity@inf.u-szeged.hu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Code to reproduce our results can be found at `https://github.com/szegedai/robust-stitching`

- We show that robust and non-robust representations are functionally compatible regarding accuracy. In other words, despite all the differences, these representations can be stitched together while preserving classification accuracy.
- However, when stitching robust representations to either robust or non-robust representations to maximize accuracy, robust accuracy decreases significantly.
- We introduce adversarially robust stitching, where we train the stitching transformation using adversarial training. We show that two robust representations can be matched using robust stitching while keeping both accuracy and robust accuracy.
- However, robust and non-robust representations cannot be matched in general using robust stitching either; no meaningful robustness can be maintained in general, but there is one exception: when the front network is robust and stitching is done in the layers close to the output. (To a lesser extent, matching is possible also when the end network is robust and stitching is done very close to the input.)
- We argue that this somewhat surprising exception is due to the fact that representations are already clustered towards the last layers in both robust and non-robust networks, which makes them much easier to match for the stitching transformation.
- We provide some more support for this hypothesis through demonstrating that even cross-task stitching is successful when done in the last layers, and the stitched network remains invertible in spite of replacing its last few layers from a non-robust donor network. This suggests that the last layers perform the same function in both robust and non-robust networks.

1.2. Related work

Here, we focus on results that are directly related to our research. These can be classified into two areas: model stitching and comparing robust representations.

Model stitching. Among many other techniques, (Lenc & Vedaldi, 2019) introduce the concept of 1×1 convolutional *stitching layers* to connect two different "half networks" by transforming the representation of the front network to the representation of the end network. They showed that the accuracy of the resulting stitched network can approach the accuracy of the original donor networks. They also found that the early, preprocessing layers of networks trained on different tasks are interchangeable. The similarity of the feature representations (that is, the layer before the final logit layer) was studied by (McNeely-White et al., 2020). Expanding on these works, (Csiszárik et al., 2021) demonstrated that model stitching can be used to study the *functional similarity* of representations by answering the following question: *can a network achieve its task using the representation of another network*? They, along with (Bansal et al., 2021) showed that networks trained on the same task but under different settings can be stitched with minimal loss of accuracy and thus all successful networks seem to learn functionally similar representations.

Comparing robust representations. The work of (Cianfarani et al., 2022) shares very similar goals to ours, but they study similarity with the help of the centered kernel alignment (CKA) representational similarity metric by (Kornblith et al., 2019), which limits their ability to disentangle functional aspects of similarity. They show that the representations of adversarial inputs are similar to those of benign inputs in the early layers even in non-robust networks. This suggests that adversarial attacks exert their effect towards deeper layers. However, (Davari et al., 2022) showed that the representations in early layers can produce high CKA values between networks with drastically different functionalities. Moreover, several studies have demonstrated various inadequacies of popular representational similarity metrics regarding their interpretability and their ability to capture the functional properties of representations (Ding et al., 2021; Csiszárik et al., 2021; Mirzadeh et al., 2021; Bansal et al., 2021). (Jones et al., 2022) study model stitching of robust models at the penultimate (feature) layer. However, they do not consider other layers, or the compatibility of robust and non-robust representations, and, most importantly, they do not study robust accuracy either, a key functional aspect of robust representations.

2. Model Stitching

Our notion of network stitching closely follows that of (Csiszárik et al., 2021). Here, we summarize the basic concepts and propose an adversarially robust variant of stitching as well.

2.1. Notation

Let $f : \mathcal{X} \to \mathcal{Y}$ be a feedforward neural network composed of m layers: $f = f_m \circ \cdots \circ f_1$, where the function $f_i : \mathcal{A}_{f,i-1} \to \mathcal{A}_{f,i}$ maps the activation space of layer $i-1$ to the activation space of layer i. By definition, $\mathcal{A}_{f,0} = \mathcal{X}$.

Since stitching will act on two "half networks", we introduce the notations $f_{\leq l} = f_l \circ \cdots \circ f_1$ and $f_{>l} = f_m \circ \cdots \circ f_{l+1}$. Clearly, $f = f_{>l} \circ f_{\leq l}$ and for an input $x \in \mathcal{X}$, $f_{\leq l}(x) \in \mathcal{A}_{f,l}$.

2.2. The Abstract Stitching Problem

The goal of stitching is to find out whether two given frozen networks $f, g : \mathcal{X} \to \mathcal{Y}$ have functionally compatible inter-

nal representations. To examine this, we select layer i from network f and layer j from network g, and we wish to find a transformation $T : \mathcal{A}_{f,i} \to \mathcal{A}_{g,j}$ such that the composition $g_{>j} \circ T \circ f_{\leq i}$ (called the Frankenstein network, or stitched network) is functionally similar to g or f. Depending on how we formulate the constraints on transformation T and on how we define functional similarity, this framework allows for a multitude of interesting avenues for investigating network representations. In the following, we discuss our specific framework.

Requirements on T. Transformation T should not increase the capacity of the model significantly, but at the same time it should be expressive enough to allow for non-trivial mappings. The consensus is that affine mappings are suitable (Csiszárik et al., 2021; Bansal et al., 2021). For convolutional layers, we adopt the proposal of (Csiszárik et al., 2021). Here, the activation spaces take the form of $\mathbb{R}^{w \times h \times c}$, where c is the number of feature maps of width w and height h. The transformation T is implemented by 1×1 convolution layers (including bias terms) for each of the c feature maps, thus applying the same affine mapping $M : \mathbb{R}^c \to \mathbb{R}^c$ at each of the $w \times h$ positions.

Functional similarity. Here, we focus on the similarity of a given metric, in our case accuracy or robust accuracy, over a supervised classification problem, given by an underlying distribution $p(x, y)$ over $\mathcal{X} \times \mathcal{Y}$.

2.3. Stitching for Accuracy

When similarity is defined based on classification accuracy, to find the best transformation T, we solve the learning task

$$\arg\min_{\theta} \mathbb{E}_{p(x,y)}[\mathcal{L}([g_{>j} \circ T_\theta \circ f_{\leq i}](x), y)] \tag{1}$$

using a suitable surrogate loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ from the distribution $p(x, y)$. We train only the parameters θ of T_θ while freezing $g_{>j}$ and $f_{\leq i}$.

A word of caution. Note, that this is an *indirect way* to achieve functional similarity to either f or g. In principle, the stitched network can outperform both f and g, in which case we know that the chosen function class implementing the transformation T is too powerful, or f and g are not optimal. However, when the stitched network underperforms both f and g then the representations are most likely incompatible with respect to the chosen transformation class, except if the optimization of the stitching layer fails (for example, if the chosen surrogate loss $\mathcal{L}$ is inappropriate, or the hyperparameters are incorrect, etc.).

2.4. Stitching for Robust Accuracy

Here, functional similarity is defined based on robust accuracy, so we should optimize robust accuracy while stitching. To this end, we shall use the adversarial version of our learning task following the strategy of (Madry et al., 2018; Goodfellow et al., 2015), given by

$$\begin{aligned}\arg\min_{\theta} \mathbb{E}_{p(x,y)}[(1-\alpha)\mathcal{L}([g_{>j} \circ T_\theta \circ f_{\leq i}](x), y) + \\ \alpha \max_{\delta \in \mathcal{S}} \mathcal{L}([g_{>j} \circ T_\theta \circ f_{\leq i}](x+\delta), y)],\end{aligned} \tag{2}$$

where $\mathcal{S} = \{\delta : \|\delta\|_p \leq \epsilon\}$ is the set of possible perturbations. We will work only with ℓ_∞ norm perturbations ($p = \infty$) here. Parameter $\alpha \in [0, 1]$ defines the proportion of adversarial training samples. In Appendix B, we present results with alternative adversarial training strategies as well.

2.5. Functional Similarity is Fine-Grained

We wish to emphasize that stitching always requires a specific definition of functional similarity, in other words, the concept of similarity is not meaningful without referring to a specific functional property such as a performance metric. Therefore, it is logically possible that two representations are similar according to one functional property, but very different according to another one. This allows for discussing similarity in a *more fine-grained* manner than function-independent similarity metrics allow. Here, we discuss the case of two metrics: accuracy and robust accuracy.

3. Stitching Robust Networks for Accuracy

In our first set of experiments, we stitch the representations of robust networks to robust as well as non-robust networks using accuracy as a similarity metric. We find that *robust and non-robust networks are functionally similar regarding accuracy*, independently of which layer is stitched. This is somewhat unexpected, based on the intuition that robust and non-robust representations are known to be qualitatively different, as we argued previously (see also Figure 9).

We also find that stitching a robust network to itself, that is, inserting an affine layer and optimizing it for plain accuracy, can *reduce the robustness of the network significantly while increasing plain accuracy*. This result would not be surprising if we were fine-tuning the entire network, however, we can shift the robustness-accuracy tradeoff significantly by training just a very small set of parameters defining the inserted affine transformation, while freezing the rest of the network.

Experimental Setup. Throughout the paper, we present experimental results over the CIFAR-10 dataset (Krizhevsky, 2009) using a number of pre-trained ResNet-18 (He et al., 2016b) networks, and each experiment is repeated three times independently. For additional datasets and architectures please refer to Appendix B. We experiment with two robust networks, both available from Robust-

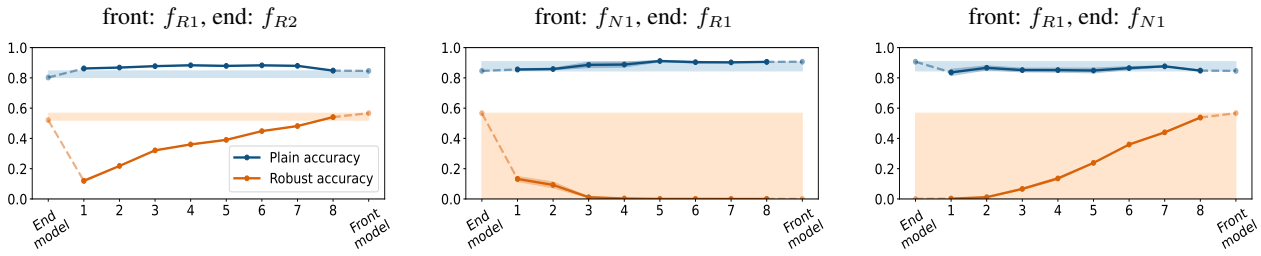

Figure 1. Stitching plots for the indicated donor networks, stitching optimized for plain accuracy.

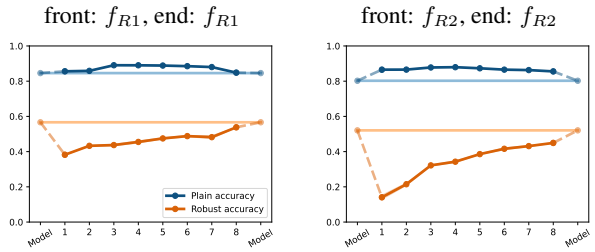

Figure 2. Stitching plots for identical front and end networks, stitching optimized for plain accuracy.

Bench (Croce et al., 2021): f_{R1} denotes the ResNet-18 network of (Sehwag et al., 2022) and f_{R2} is the network of (Addepalli et al., 2021). The network f_{N1} is a basic non-robust network that we trained ourselves. We solve the stitching problem given in Equation (1). For the additional details of our experimental settings see Appendix A.

The stitching plot. Our main tool for presenting our results is the stitching plot (see, for example, Figure 1). On the horizontal axis, we indicate the location of the stitching layer within the stitched network sorted according to increasing distance from the input. The first point indicates the full end model (as if the stitching was before the network, thus combining zero layers from the front network and the complete end network), and the last point indicates the full front model. The rest of the points correspond to the basic blocks of the ResNet-18 architecture, sorted according to increasing distance from the input. Stitching layers are inserted after the basic blocks, after the skip connection (see Appendix A.5 for more details).

On the vertical axis, we indicate accuracy and robust accuracy of both donor networks in the following way: the interval defined by the plain accuracy of the front and end networks is indicated in blue, and the interval defined by robust accuracy is indicated in yellow. We evaluate every stitched network for accuracy and robust accuracy, and these values are plotted. Robust accuracy is evaluated using AutoAttack (Croce & Hein, 2020) (see Appendix A.4).

3.1. Experimental Results

Plain accuracy. Figure 1 illustrates the results for the two possible combinations of robust and non-robust networks, and for stitching two robust networks. The most obvious observation is that *representation matching is successful* in all the cases, in that the stitched networks either interpolate the plain accuracy of the two donor networks, or even surpass it in the case of stitching two robust networks.

Robust accuracy. Although, here, stitching is optimized for plain accuracy, we show the robust accuracy of the stitched network as well, which shows interesting patterns. When the front network is non-robust, the robustness of the stitched network is completely lost after the first few layers. However, when the end network is non-robust, interestingly, robust accuracy is preserved in the last layers. Even more interestingly, we see a similar pattern in the case of stitching two robust networks, where robustness is also lost in the first half of the network. Clearly, the preservation of robustness depends on the location of the stitching. In Section 5 we will argue that this phenomenon could be due to the clustering of representations that increases towards the output.

Self-stitching. For any similarity measure, an important sanity check is whether a representation is considered similar to itself. Figure 2 illustrates this sanity check for two robust networks. It is striking that, at almost any layer, the accuracy of the stitched network is higher than that of the original network. Considering that the transformation has a low complexity, this indicates that the robust networks are far from optimal w.r.t. plain accuracy, as they can be significantly improved by just a weak extra transformation layer.

In general, the result is very similar to the case of stitching two different robust networks. This means that the observed shift in the accuracy-robustness tradeoff is not due to incompatibilities of the representation of robust features (as they are identical here), instead, it is caused by the simple affine stitching transformation. This is a potentially useful observation for the study of the accuracy-robustness tradeoff problem and represents an avenue for future research.

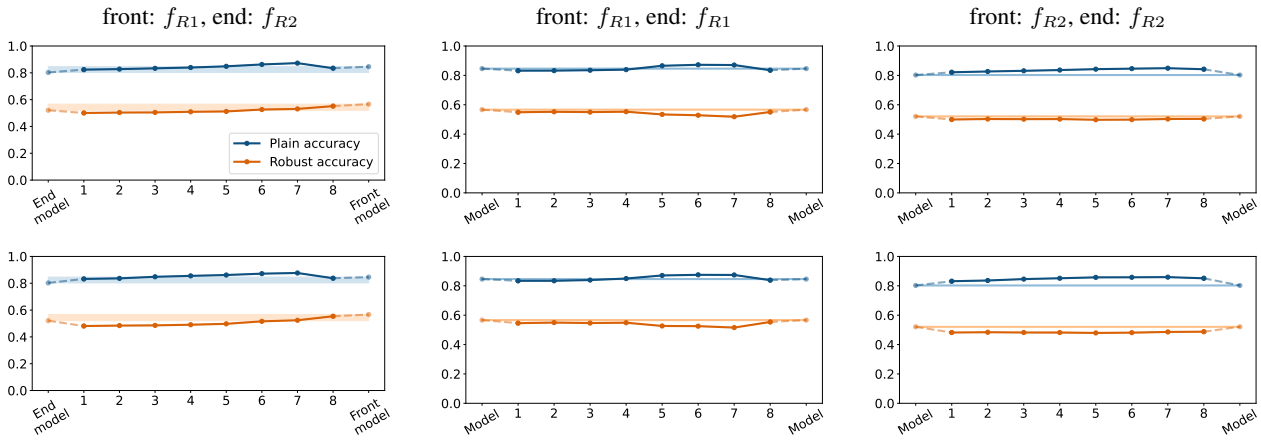

Figure 3. Stitching plots for the indicated donor networks, stitching optimized for robust accuracy. Trained using 100% (top row) and 50% (bottom row) adversarial examples (that is, with $\alpha = 1$ and $\alpha = 0.5$, respectively).

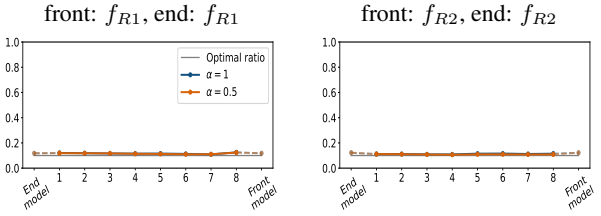

Figure 4. The ratio of the most often predicted class over the test set of the stitched networks trained using 100% ($\alpha = 1$) and 50% ($\alpha = 0.5$) adversarial examples. Note that the optimal value of this ratio is $1/10$ (indicated in black), because CIFAR-10 has 10 classes with equal probabilities.

4. Adversarial Stitching

Here, we use the same experimental setup as in Section 3, except we now solve the learning task defined in Equation (2), and we use an additional non-robust model f_{N2}.

Our main finding is that *the representations of two robust networks can be matched, preserving both robust and plain accuracy*. However, *matching robust and non-robust representations is a lot harder*, and the success of the matching strongly *depends on the location of the matched layer*. After presenting the experimental results, we shall offer a speculative explanation that is based on the clustering of the representations in Section 5.

4.1. Robust-Robust Stitching

Figure 3 presents the case when we stitch two robust networks together. Clearly, the robust representations are compatible. It is also remarkable that the results with $\alpha = 1$ and $\alpha = 0.5$ are practically identical. A possible explanation is that the adversarial training samples generate a larger loss value (since there, the loss is explicitly maximized) and therefore they dominate the learning task.

It is remarkable that in the self-stitching cases we can still see a slight shift in the robustness-accuracy tradeoff, although to a much lesser extent than in the case of non-robust stitching. Note that the two robust networks we examine were trained with sophisticated methods well beyond the adversarial training we apply here, and, roughly speaking, stitching may "undo" some of the effects of those techniques.

4.2. Plain-Plain Stitching

Figure 5 includes our results with stitching two non-robust networks. It is not expected that the stitched network can be made robust just by adding the stitching layer and training it adversarially, but it is still interesting to study this case to gain some insights. Indeed, with $\alpha = 1$ the matching completely fails. Although we can see a non-zero robust accuracy, this is due to the matching going awry. The stitched network classifies most inputs to a single class in a desperate attempt to achieve a robust accuracy of $1/10$ this way (see Figure 6). Hence, plain accuracy is also very low.

However, with $\alpha = 0.5$ we can see an interesting difference. Here, when stitching by the layers closer to the output, we can see that the plain accuracy is recovered, despite training for robust accuracy, which stays close to zero. Clearly, due to using clean inputs as well, a good stitching transformation with respect to plain accuracy is also a local optimum for the robust loss in layers closer to the output, but this is not true in earlier layers. The data in Figure 6 also supports this interpretation.

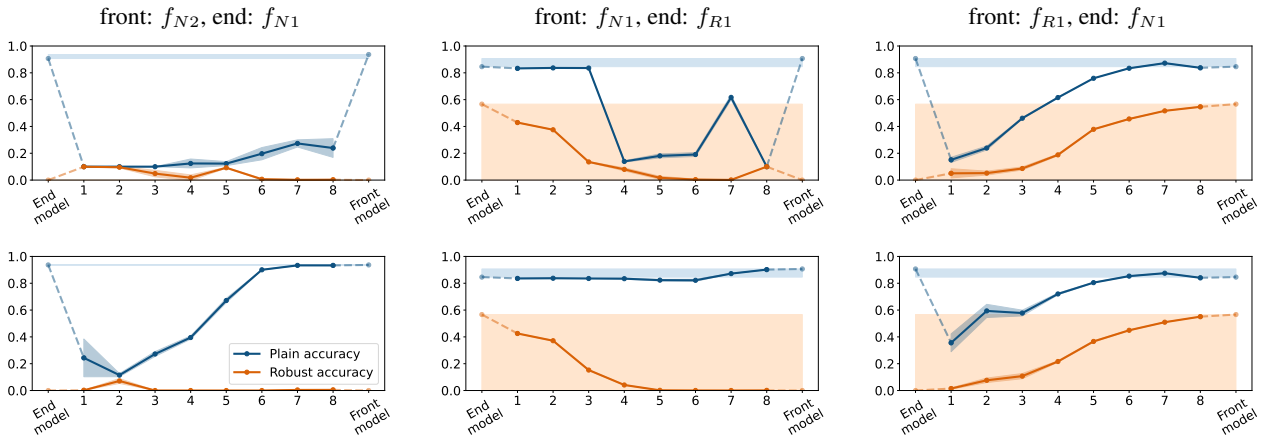

Figure 5. Stitching plots for the indicated donor networks, stitching optimized for robust accuracy. Trained using 100% (top row) and 50% (bottom row) adversarial examples (that is, with $\alpha = 1$ and $\alpha = 0.5$, respectively).

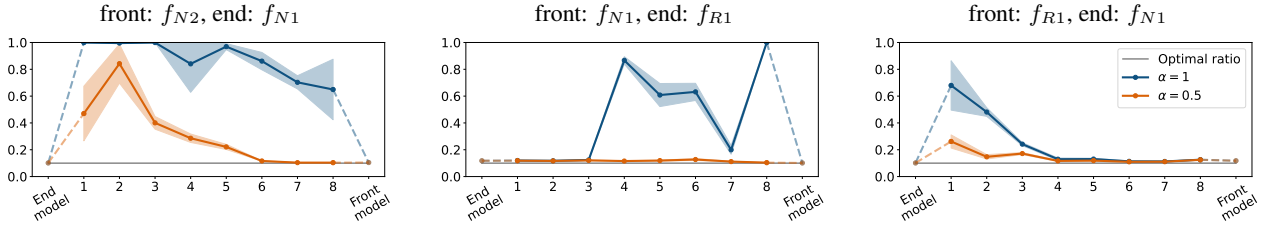

Figure 6. The ratio of the most often predicted class over the test set of the stitched networks trained using 100% ($\alpha = 1$) and 50% ($\alpha = 0.5$) adversarial examples. Note that the optimal value of this ratio is $1/10$ (indicated in black), because CIFAR-10 has 10 classes with equal probabilities.

4.3. Plain-Robust Stitching

Let us now have a look at the case with a non-robust (plain) front network and a robust end network. Here, in the first preprocessing layers some robustness can be recovered in a non-degenerate manner, that is, we can preserve the plain accuracy (see also Figure 6). This indicates that the first few non-robust layers still preserve some of the robust features that can be mapped to the robust representation of the end network.

However, from the middle of the network the stitching plot indicates a very similar behavior to the plain-plain case we discussed above: with $\alpha = 1$ we get degenerate stitching layers, and with $\alpha = 0.5$ plain accuracy is preserved but robustness vanishes (that is, what we get is essentially plain stitching despite robust training). This indicates that the non-robust front network can no longer provide any robust features. Also, the robust accuracy is very similar with both $\alpha = 1$ and $\alpha = 0.5$, which indicates that $\alpha = 0.5$ is a better choice in this case.

4.4. Robust-Plain Stitching

The last remaining case involves a robust front network and a non-robust end network. In this case, the first half resembles the plain-plain case, when we can see degenerate mappings with $\alpha = 1$, indicating that the first few preprocessing layers cannot be matched meaningfully to the representation of the non-robust end network. Also, setting $\alpha = 0.5$ changes the stitching transformation to favor plain accuracy significantly more, although here, plain accuracy cannot be recovered completely.

However, in the second half of the network, something interesting happens. Here, the last few layers of the plain end network are perfectly suitable as a replacement for the robust network's last layers. In other words, the first layers are a lot less interchangeable than the last layers. It is also remarkable that the robustness is, again, very similar with $\alpha = 1$ and $\alpha = 0.5$, while the plain accuracy is very clearly better with $\alpha = 0.5$. This suggests that $\alpha = 0.5$ is a better choice for adversarially robust stitching.

4.5. Concluding Remarks

As a general observation, we can conclude that the last layers appear to be interchangeable, in the sense that the properties of every type of front network can be recovered with an appropriate stitching method. However, the front layers are not always interchangeable, in the sense that the robustness of the end network quickly vanishes when non-robust preprocessing layers are added. But, from the point of view of plain accuracy, even the preprocessing layers are interchangeable.

Also, the most aggressive setting of $\alpha = 1$ causes the formation of degenerate stitching layers at locations where matching the representations is hard, as illustrated by Figure 6. Figure 4, however, indicates that in the robust-robust case matching is easy even with $\alpha = 1$.

5. The Role of Representation Clustering

The observations in Section 4.5 beg the question of what the most important difference is between early and late layers that could explain the difference in their interchangeability. A possible explanation is that this difference lies in the different clustering of the representations, at least in the case of classification problems. In general, the last layers of networks are expected to depend more on the abstract task type, while early layers depend more on the input distribution. This phenomenon is quite well understood and the role of clustering has already been identified (Goldfeld et al., 2019). This could explain why the layers close to the output are more interchangeable: these layers are similar because in each network (robust or not) the task is still to classify the input into a number of classes.

Figure 7 shows the visualizations of the representations in layers 4 to 8 using t-SNE (van der Maaten & Hinton, 2008). Indeed, in both the robust and non-robust networks we can observe a clear increase in the clustering of the representations.

5.1. Cross-Task Stitching

If the final layers of the network are indeed abstract and depend less on the input distribution then the last layers should be interchangeable even if the donor networks were trained on different datasets, as long as the abstract task type is compatible.

We tested this hypothesis via stitching networks that were trained on different tasks: CIFAR-10 and the SVHN dataset. Both tasks require the network to classify the input into 10 classes, but the input distribution is rather different. The CIFAR-10 networks we used were our usual f_{N1} and f_{R1}, and we also trained an SVHN network g_{N1} using non-adversarial training. We used plain stitching based on Equation (1). For more details, please refer to Appendix A.

In Figure 8 we can see that our hypothesis is supported, in that the last layers are indeed interchangeable. That is, if we train the stitching layer on the dataset of the front network then replacing the last few layers is possible while preserving accuracy on the task of the front network.

Also, note that our findings complement the results of (Terzi et al., 2021), where the authors study transferability by building a linear classifier over the feature representation of a model and show that robust representations transfer better. Here, cross-task stitching is also more successful in the last layers if the front model is robust, suggesting a richer feature representation.

We can also see that, when the stitching layer is trained over the dataset of the end network, the preprocessing layers are also interchangeable. In this case, it is also better to have a robust network as an end network.

6. Model Inversion

A characteristic property of robust networks is that they are invertible (Engstrom et al., 2019b), which suggests that robust models keep more information about the input, which is counter-intuitive (Terzi et al., 2021), given that the last layers are more abstract and, as we saw, more interchangeable, even between networks that were trained on different input distributions.

Here, we test the invertibility of several stitched networks, to see whether robust accuracy is sufficient to characterize the preservation of information. For simplicity, we define the inversion task assuming our setup and notation, where the feature representation of a ResNet-18 network f is $f_{\leq 8}$. Under these assumptions, the inverse of a given feature representation $z \in \mathcal{A}_{f,8}$ is computed by solving the optimization problem

$$f_{\leq 8}^{-1}(z) = \arg\min_x \|f_{\leq 8}(x) - z\|_2. \tag{3}$$

Figure 9 illustrates the invertibility of several networks, by taking a CIFAR-10 image x and computing $f_{\leq 8}^{-1}(f_{\leq 8}(x))$ for a given network f. For our detailed setup and more inversion results please see Appendix C.

The striking observation is that the stitched network in which the last layers are donated by a non-robust network is invertible, in fact, perhaps better than the full robust network. It is clear that the *detailed information about the input is preserved while passing through the last non-robust layers*. This also supports the hypothesis that the last layers have less to do with robustness and more to do with the abstract task at hand.

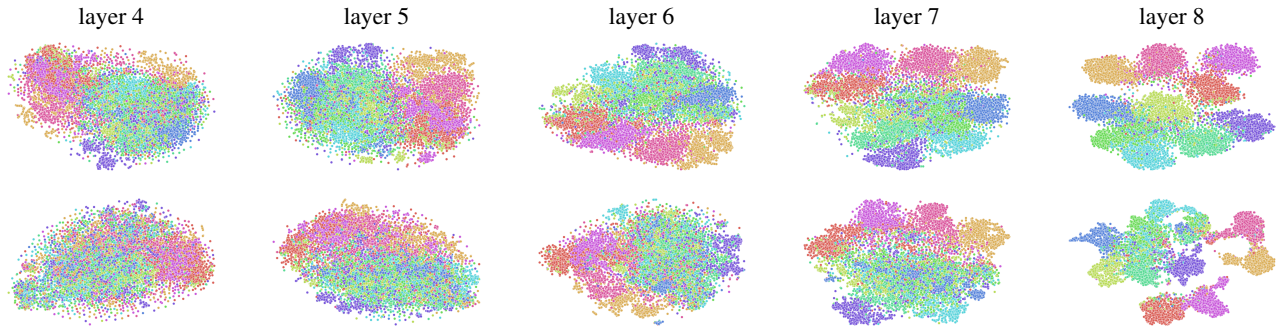

Figure 7. The t-SNE visualization of various layers of f_{N1} (top row) and f_{R1} (bottom row). Each point is a sample and the color represents one of 10 classes.

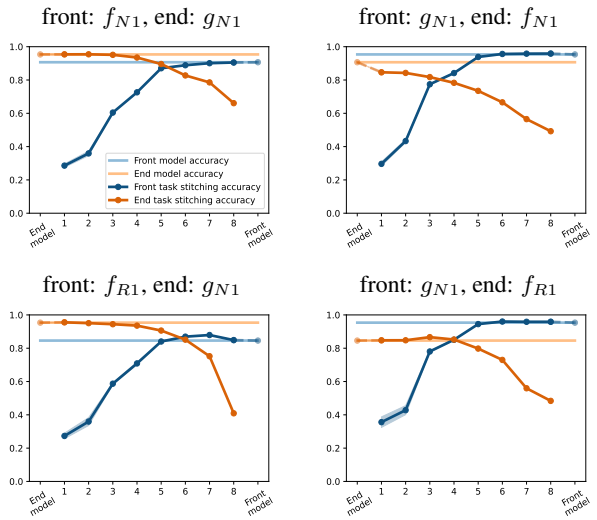

Figure 8. Cross-task stitching using CIFAR-10 networks (f_{N1}, f_{R1}) and an SVHN network (g_{N1}). The curves show the plain accuracy of the stitched networks when stitching over CIFAR-10 and SVHN.

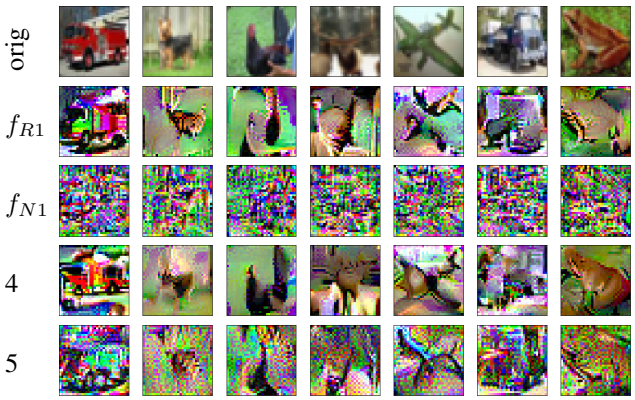

Figure 9. Inverted feature representations of several networks over a random sample of CIFAR-10 images (top row). In row 4 we have $f_{N1,>6} \circ T_{rob} \circ f_{R1,\leq 6}$, that is, the front network is f_{R1}, the end network is f_{N1}, robust stitching with $\alpha = 1$ at layer 6. Row 9 shows $f_{R1,>2} \circ T_{rob} \circ f_{N1,\leq 2}$ using robust stitching with $\alpha = 1$.

7. Limitations

Since we study stitching using adversarial training, the experiments are very expensive which limited the number of experiments we could complete and the complexity of the models and the datasets we could work with, given the amount of computational resources we have access to. At present, this work represents about one GPU year's worth of computation (not counting the experiments in our path-finding phase). It would be very useful to study several other datasets and network architectures beyond what has been presented in the paper and in the Appendix to support the generality of our empirical findings better.

8. Conclusions

Our main conclusion is that robust representations are functionally similar to non-robust representations in terms of accuracy but, in general, they are not similar in terms of robust accuracy.

However, the last layers of any two networks are interchangeable, in the sense that the stitched network is similar to the front network, even if the front network is robust and the end network is not, and even if the two stitched networks were trained on different tasks of the same type.

We argued that the clustering of the representations, that happens independently of robustness and input distribution, is a possible explanation of why the last layers are interchangeable.

These conclusions are interesting from several points of view. First of all, they underline the fact that representation similarity is fine-grained, that is, similarity is ill-defined

without referring to function.

Second, our experiments provide independent evidence that robust and non-robust representations might indeed be different, and we can even localize the layers where the difference is the largest: they are in the transition phase between preprocessing and clustering (or semantic) layers, where preprocessing is already high level but the semantic structure is not yet mature.

Acknowledgements

This work was supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory and project TKP2021-NVA-09, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme and the ÚNKP-22-2-SZTE-343 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. We are grateful for the possibility to use the ELKH Cloud (Héder et al., 2022), which helped us achieve the results published in this paper.

References

Addepalli, S., Jain, S., Sriramanan, G., Khare, S., and Radhakrishnan, V. B. Towards achieving adversarial robustness beyond perceptual limits. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. URL https://openreview.net/forum?id=SHB_znlW5G7.

Bai, Y., Yan, X., Jiang, Y., Xia, S., and Wang, Y. Clustering effect of adversarial robust models. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 29590–29601, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f770b62bc8f42a0b66751fe636fc6eb0-Abstract.html.

Bansal, Y., Nakkiran, P., and Barak, B. Revisiting model stitching to compare neural representations. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 225–236, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/01ded4259d101feb739b06c399e9cd9c-Abstract.html.

Cianfarani, C., Bhagoji, A. N., Sehwag, V., Zhao, B., Zheng, H., and Mittal, P. Understanding robust learning through the lens of representation similarities. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://openreview.net/forum?id=SbAaNa97bzp.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning, (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216. PMLR, 2020. URL http://proceedings.mlr.press/v119/croce20b.html.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/a3c65c2974270fd093ee8a9bf8ae7d0b-Abstract-round2.html.

Csiszárik, A., Kőrösi-Szabó, P., Matszangosz, Á. K., Papp, G., and Varga, D. Similarity and matching of neural network representations. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 5656–5668, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/2cb274e6ce940f47beb8011d8ecb1462-Abstract.html.

Davari, M., Horoi, S., Natik, A., Lajoie, G., Wolf, G., and Belilovsky, E. On the inadequacy of CKA as a measure of similarity in deep learning. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=rK841rby6xc.

Ding, F., Denain, J.-S., and Steinhardt, J. Grounding representation similarity through statistical testing. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 1556–1568. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/0c0bf917c7942b5a08df71f9da626f97-Paper.pdf.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019a. URL https://github.com/MadryLab/robustness.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Tran, B., and Madry, A. Adversarial robustness as a prior for learned representations. Technical Report 1906.00945,

arXiv.org, 2019b. URL https://arxiv.org/abs/1906.00945.

Goldfeld, Z., van den Berg, E., Greenewald, K. H., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. Estimating information flow in deep neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2299–2308. PMLR, 2019. URL http://proceedings.mlr.press/v97/goldfeld19a.html.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6572.

Gowal, S., Rebuffi, S., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. Improving robustness using generated data. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 4218–4233, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/21ca6d0cf2f25c4dbb35d8dc0b679c3f-Abstract.html.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016a. doi: 10.1007/978-3-319-46493-0_38. URL https://doi.org/10.1007/978-3-319-46493-0_38.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016b. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

Héder, M., Rigó, E., Medgyesi, D., Lovas, R., Tenczer, S., Török, F., Farkas, A., Emődi, M., Kadlecsik, J., Mező, G., Pintér, Á., and Kacsuk, P. The past, present and future of the ELKH cloud. *Információs Társadalom*, 22 (2):128, aug 2022. doi: 10.22503/inftars.xxii.2022.2.8. https://science-cloud.hu/.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 125–136. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8307-adversarial-examples-are-not-bugs-they-are-features.pdf.

Jones, H., Springer, J. M., Kenyon, G. T., and Moore, J. If you've trained one you've trained them all: Inter-architecture similarity increases with robustness. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022. URL https://openreview.net/forum?id=BGfLS_8j5eq.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. Similarity of neural network representations revisited. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

LeCun, Y., Cortes, C., and Burges, C. MNIST handwritten digit database, 2010. URL http://yann.lecun.com/exdb/mnist.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127(5):456–476, 2019. doi: 10.1007/s11263-018-1098-y. URL https://doi.org/10.1007/s11263-018-1098-y.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

McNeely-White, D., Beveridge, J. R., and Draper, B. A. Inception and resnet features are (almost) equivalent. *Cognitive Systems Research*, 59:312–318, 2020. doi: https://doi.org/10.1016/j.cogsys.2019.10.004. URL https://www.sciencedirect.com/science/article/pii/S1389041719305066.

Mirzadeh, S., Farajtabar, M., Görür, D., Pascanu, R., and Ghasemzadeh, H. Linear mode connectivity in multitask and continual learning. In *9th International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Fmg_fQYUejf.

Morcos, A. S., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 5732–5741, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a7a3d70c6d17a73140918996d03c014f-Abstract.html.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., and Edwards, B. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018. URL https://arxiv.org/pdf/1807.01069.

Rade, R. and Moosavi-Dezfooli, S.-M. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. URL https://openreview.net/forum?id=BuD2LmNaU3a.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 6076–6085, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., and Mittal, P. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=WVX0NNVBBkV.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations (ICLR)*, 2015. URL http://arxiv.org/abs/1409.1556.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations (ICLR)*, 2014. URL http://arxiv.org/abs/1312.6199.

Terzi, M., Achille, A., Maggipinto, M., and Susto, G. A. Adversarial training reduces information and improves transferability. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021*, pp. 2674–2682. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16371.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P. (eds.), *Proceedings of the British Machine Vision Conference 2016, (BMVC)*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7472–7482. PMLR, 2019. URL http://proceedings.mlr.press/v97/zhang19p.html.

A. Experimental Setup

A.1. Datasets and Architectures

Our experiments were conducted using a number of models trained on the following datasets: MNIST (LeCun et al., 2010), Fashion MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky, 2009) and SVHN (Netzer et al., 2011). We used the single-class version of the SVHN dataset and the standard version available in PyTorch for all the other datasets. All the input image color values during training and evaluation were scaled into the $[0, 1]$ interval. The model architectures we used include VGG-11 (Simonyan & Zisserman, 2015), ResNet-18, ResNet-50 (He et al., 2016b), PreActResNet-18 (He et al., 2016a) and WideResNet-28-10 (Zagoruyko & Komodakis, 2016). Table 1 contains the details of all the models used in our experiments.

Table 1. Accuracy and robust accuracy (evaluated by the method explained in A.4) of the models used in our experiments. Column *Source* indicates the publication in the case of state-of-the-art robust networks and the robust training method (AT: (Madry et al., 2018), TRADES: (Zhang et al., 2019)) in the case of our own robust networks. Column *Rob. Acc.* is measured as described in Appendix A.4.

Network	Architecture	Dataset	Accuracy	Rob. Acc.	Source
h_{N1}	VGG-11	MNIST	99,61%	0%	
h_{N2}	VGG-11	Fashion MNIST	94,12%	0%	
h_{N3}	ResNet-18	MNIST	99,97%	0%	
h_{N4}	ResNet-18	Fashion MNIST	94,77%	0%	
g_{N1}	ResNet-18	SVHN	95,36%	0%	
g_{N2}	VGG-11	SVHN	95,35%	0%	
f_{N1}	ResNet-18	CIFAR10	90,67%	0%	
f_{N2}	ResNet-18	CIFAR10	93,66%	0%	
f_{N3}	PreActResNet-18	CIFAR10	93,82%	0%	
f_{N4}	ResNet-50	CIFAR10	94,47%	0%	
f_{N5}	WideResNet-28-10	CIFAR10	92,91%	0%	
f_{N6}	VGG-11	CIFAR10	90,23%	0%	
h_{R1}	ResNet-18	MNIST	99,39%	94,10%	AT, PGD, $\alpha = 1$
h_{R2}	ResNet-18	Fashion MNIST	90,65%	45,40%	AT, PGD, $\alpha = 1$
g_{R1}	ResNet-18	SVHN	85,32%	52,15%	TRADES, $\beta = 10$
g_{R2}	ResNet-18	SVHN	91,71%	44,48%	AT, $\alpha = 0.5$
f_{R1}	ResNet-18	CIFAR10	84,59%	56,66%	(Sehwag et al., 2022)
f_{R2}	ResNet-18	CIFAR10	80,24%	52,07%	(Addepalli et al., 2021)
f_{R3}	ResNet-18	CIFAR10	79,97%	47,50%	TRADES, $\beta = 10$
f_{R4}	PreActResNet-18	CIFAR10	89,02%	58,34%	(Rade & Moosavi-Dezfooli, 2021)
f_{R5}	ResNet-50	CIFAR10	87,03%	51,53%	(Engstrom et al., 2019a)
f_{R6}	WideResNet-28-10	CIFAR10	87,50%	64,31%	(Gowal et al., 2021)

A.2. Hyperparameters for Training Donor Models

Our hyperparameter settings for model training closely followed those of (Csiszárik et al., 2021). All of our non-robust models were trained minimizing the cross-entropy loss function with an ℓ_2 weight decay coefficient of 10^{-4} using the stochastic gradient descent (SGD) optimizer with a Nesterov momentum of 0.9. For better generalization we used the following augmentation techniques: random horizontal flip, random crop and in the case of MNIST random affine transformations with scaling in the range of $[0.9, 1.1]$ and at most 5 degrees of rotation and shearing. We set the initial learning rate 10^{-3} when training VGG-11 models and 10^{-1} for every other architecture. During training, we divided the learning rate by 10 at 1/3 and 2/3 of the total number of training epochs. We trained the MNIST models for 30 epochs and the CIFAR-10, SVHN and Fashion MNIST models for 200 epochs.

A.3. Adversarial Training

We trained our robust models with the ℓ_∞ threat model with the standard setting $\epsilon = 0.3$ for the MNIST and Fashion MNIST models and $\epsilon = 8/255$ for the CIFAR-10 and SVHN models. For the internal maximization in adversarial training, when

using the strategy of (Madry et al., 2018; Goodfellow et al., 2015), we used the untargeted projected gradient descent (PGD) attack as described by (Madry et al., 2018) with 10 iterations and with a step size parameter of 0.1 for MNIST and Fashion MNIST models and $2/255$ for CIFAR-10 and SVHN models.

To experiment with a more diverse set of robust models, we also used the TRADES loss (Zhang et al., 2019) for adversarial training that defines the learning task

$$\arg\min_{\theta} \mathbb{E}_{p(x,y)}[\mathcal{L}(f_\theta(x), y) + \beta \max_{\delta \in \mathcal{S}} \mathcal{L}(f_\theta(x), f_\theta(x+\delta))], \tag{4}$$

where f denotes the trained model and θ denotes the parameters of f. To maximize robustness we set β to 10.

All the hyperparameters that were not mentioned in this section were identical to those described in Appendix A.2.

A.4. Evaluating Robustness

For evaluating the robustness of the original models as well as the stitched models, we used the "rand" version of AutoAttack (Croce & Hein, 2020) with the implementation of (Nicolae et al., 2018). The "rand" version of AutoAttack contains two untargeted auto-projected gradient descent (aPGD) attacks that maximize the cross-entropy and the logit margin, respectively. Both attacks were parameterized similarly: 20 iterations, no restarts and ϵ and step size values identical to those mentioned in Appendix A.3. Evaluation was conducted in the ℓ_∞ threat model. We found that this version of AutoAttack, while being slightly weaker than the standard version, is still a suitable and significantly faster way to evaluate a network's robustness against gradient-based attacks. Table 2 shows a comparison of the strength of the AutoAttack variants.

Table 2. Comparing the version of AutoAttack we used (third column) with the standard version of AutoAttack (fourth column) by evaluating the robust accuracy of five state-of-the-art robust networks with the two attacks.

Architecture	**Publication**	**AutoAttack ("rand")**	**AutoAttack (full)**
ResNet-18	(Addepalli et al., 2021)	52,07%	51,06%
ResNet-18	(Sehwag et al., 2022)	56,66%	55,54%
PreActResNet-18	(Rade & Moosavi-Dezfooli, 2021)	58,34%	57,67%
ResNet-50	(Engstrom et al., 2019a)	51,53%	49,25%
WideResNet-28-10	(Gowal et al., 2021)	67,28%	63,44%

A.5. Stitching

Our stitching implementation is based on the implementation provided by (Csiszárik et al., 2021). The stitching transformation does not contain batch normalization, however, it does contain learnable bias terms and so it is an affine transformation. We train all the stitching layers with the Adam optimizer for 30 epochs. Learning rate scheduling and hyperparameters (except for the optimizer) are identical to those described in Appendix A.2. We used the random initialization method for the stitching layer in the main part of the paper. Here, however, we also include some results with initialization with the identity mapping.

In our experiments the stitched donor networks have identical architectures. We performed stitching at architecturally corresponding layers which, according to the sanity check proposed by (Kornblith et al., 2019) should be the most similar between networks with the same architecture. In other words in all our experiments $f_{\leq l}$ and $g_{\leq j}$ as well as $f_{>l}$ and $g_{>j}$ were architecturally identical.

A.5.1. Stitching Layer Placement

When stitching residual networks we place the stitching layers at the end of residual blocks, after the piecewise addition operation. This is important because stitching inside the residual block—while technically possible—cannot be implemented properly using a single stitching layer, because the transformation of the skip connection is also necessary. Our implementation does allow for stitching to be performed inside the residual blocks with the caveat that the skip connection is taken directly from the end network. For completeness, we mention that we performed a set of preliminary experiments (not shown here) with plain stitching inside the residual blocks that showed a comparable performance.

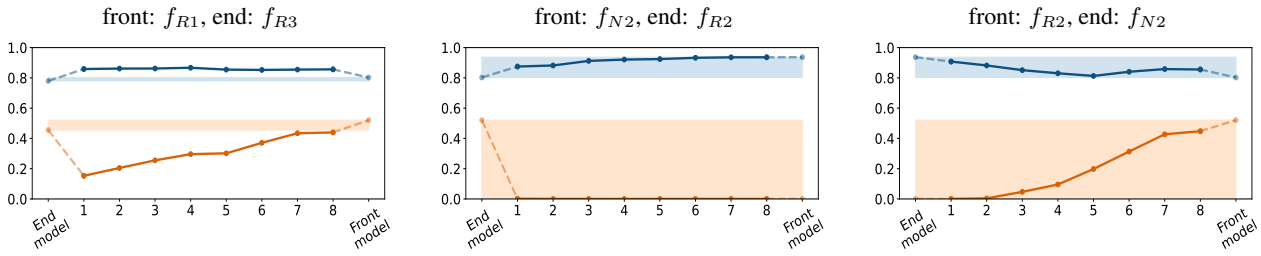

Figure 10. Stitching plots for the indicated donor networks, stitching optimized for plain accuracy. (Same as Figure 1, only with different networks.)

Another question regarding stitching layer placement is the location of the stitching layer relative to the activation function. In the case of VGG-11 and ResNet, the stitching layers are placed before the activation function in the forward order. In the case of PreActResNet and WideResNet, the stitching layers are placed after the activation function.

A.5.2. Batch Normalization

The implementation of (Csiszárik et al., 2021) allows all of the batch normalization layers' mean and variance parameters in the stitched network to be modified during the training of the stitching layer. We performed a set of preliminary experiments with plain stitching where we froze the batch normalization layers. These experiments resulted in practically identical performance, but at the cost of significantly slower convergence. Possibly to combat this, the implementation of (Bansal et al., 2021) adds one batch normalization before and after the stitching transformation. Based on our preliminary experiments, we left the batch normalization layers unfrozen, as was done in (Csiszárik et al., 2021).

A.6. Computational Resources

During our experiments we trained and evaluated over 3000 stitching layers, of which approximately 75% were trained adversarially. For our experiments we used a mixture of GeForce 2080 Ti 10G, 3060 12G, and V100DX-16C (10G virtual slice) GPUs. While plain stitching experiments took about 30 minutes for each layer on a single GPU, adversarial stitching experiments with larger networks took significantly longer. For example, the training and evaluation of each stitching layer of the ResNet-18 architecture over the CIFAR-10 dataset took approximately 3 hours on one GPU (so, one run for one full stitching plot took about a GPU-day). A layer of the ResNet-50 architecture with adversarial stitching and evaluation over the CIFAR-10 dataset took approximately 8 hours, so one run for a full stitching plot with this architecture took 124 GPU-hours, or about 5.34 days.

B. Additional Experiments

The main text of the paper focused on a fixed set of ResNet-18 networks over the CIFAR-10 dataset. Here, we present additional results with alternative networks, architectures, datasets, and algorithmic components. Here, each point of every plot corresponds to a single run of the given training algorithm.

B.1. Alternative Network Instances and Components

First, let us present experimental results using the same setting as the main text (ResNet-18 networks over the CIFAR-10 dataset). Figures 10 and 11 present results that use identical settings to the ones used in the main text, but replacing some of the networks that are stitched. The results are similar.

Apart from the method described in Section 2, we experimented with an additional adversarial stitching method as well that is based on the TRADES loss (Zhang et al., 2019):

$$\arg\min_{\theta} \mathbb{E}_{p(x,y)}[\mathcal{L}([g_{>j} \circ T_\theta \circ f_{\leq i}](x), y) + \beta \max_{\delta \in \mathcal{S}} \mathcal{L}([g_{>j} \circ T_\theta \circ f_{\leq i}](x), x + \delta)] \tag{5}$$

In our experiments, parameter β (that controls the tradeoff between accuracy and robustness) is set to 1 if at least one

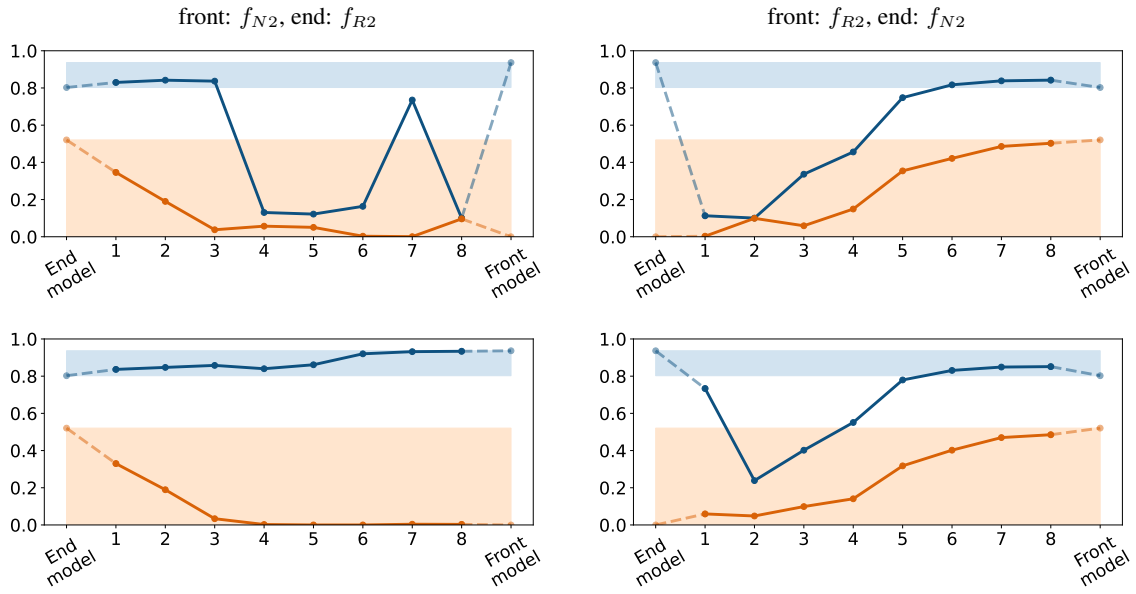

Figure 11. Stitching plots for the indicated donor networks, stitching optimized for robust accuracy. Trained using 100% (top row) and 50% (bottom row) adversarial examples. (Same as the same type of plots included in Figure 3, only with different networks.)

donor network is non-robust. This allows us to closely match the adversarial stitching method described by Equation (2) with $\alpha = 0.5$. Our experiments indicate that the two adversarial stitching methods yield similar results, as illustrated by Figure 12.

In the case of self-stitching experiments, it is an interesting question whether the stitching layer will be more similar to the identity mapping if we use the identity mapping for initialization. As Figure 13 indicates, the stitching layers appear to be very similar to the ones using random initialization.

B.2. Alternative Network Architectures

Here, still staying with the CIFAR-10 dataset, we tested additional network architectures. Figures 14 and 15 shows our stitching experiments with ResNet-50 networks, Figures 16 and 17 present the experiments with PreActResnet-18, and finally, Figures 18 and 19 present our results with WideResnet-28-10. We can conclude that similar patterns emerge in each case. One case stands out, however: the WideResnet architecture results in a better robust accuracy for non-robust stitching than the other architectures, when two robust networks are stitched together.

B.3. Alternative Datasets

Here, we present a set of results over other datasets. In particular, Figures 20 and 21 show our results over the SVHN dataset, Figure 22 contains our results with the Fashion-MNIST dataset, and Figure 23 presents stitching plots over the MNIST dataset. The patterns that we observe are, again, consistent with the previous results. Perhaps MNIST stands out slightly, because it preserves more robustness than the other networks when the front network is robust. This could be due to the fact that a lot of the adversarial vulnerability of MNIST networks originate from manipulating the background (which is constant black in all the clean examples), and robust networks tend to threshold the background, which is an easy fix for this particular attack vector.

B.4. Additional Cross-Task Experiments

We experimented with cross-task stitching using other datasets and networks as well, the results are shown in Figures 24 to 26. The results are consistent with out previous results. One observation we can make is that cross-task stitching works better when the network solving the harder task (CIFAR-10, and Fashion-MNIST, respectively) is the front network. However, here, it is also clear that the last layers are interchangeable, even if the weaker network is the front network. This

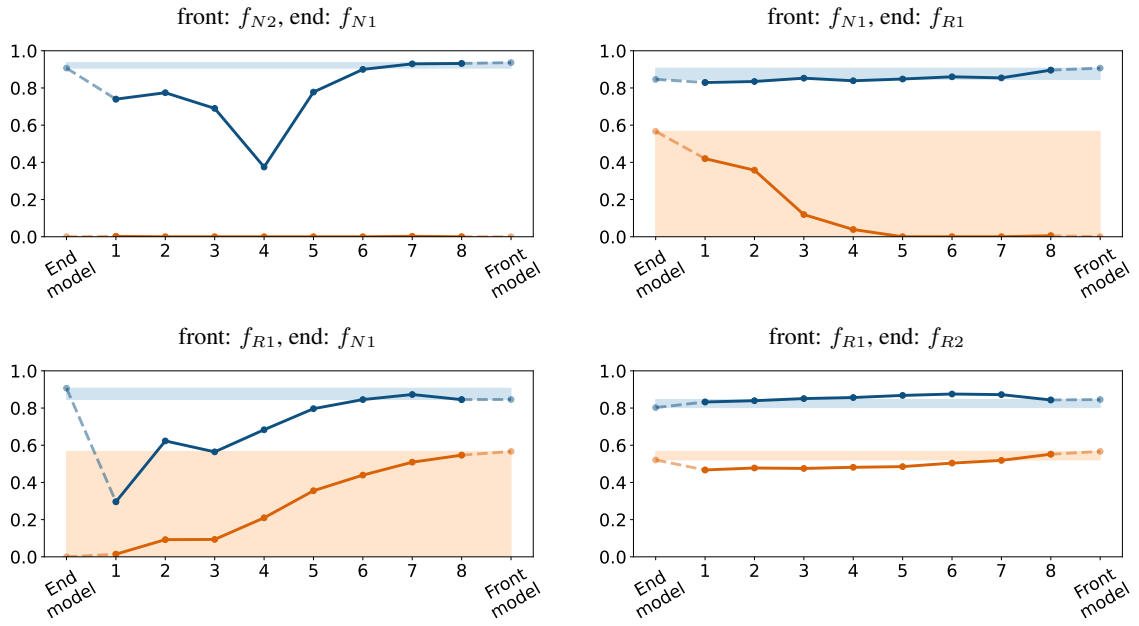

Figure 12. Stitching plots for the indicated donor networks, stitching optimized for robust accuracy using TRADES with $\beta = 1$.

is not the case for the preprocessing layers.

C. Additional Inversion Results

Our representation inversion experiments are based on the implementation provided by (Engstrom et al., 2019b) and our hyperparameter settings are identical to theirs. For optimization we use an ℓ_2-bounded PGD with random initialization, $\epsilon = 1000$ and a step size of 1.

Figure 27 shows the inverted representations of additional networks relative to those shown in Figure 9. These new networks correspond to rows 6–9. All of these new stitched networks use the robust f_{R1} as a front network. We were interested in how the invertibility depends on the end network and the stitching method. The end networks include g_{N1}, which is trained on a different task (SVHN). The figure indicates that invertibility is surprisingly robust to these factors, and the last layers seem to be interchangeable almost independently of the end network, keeping also a reasonable degree of invertibility.

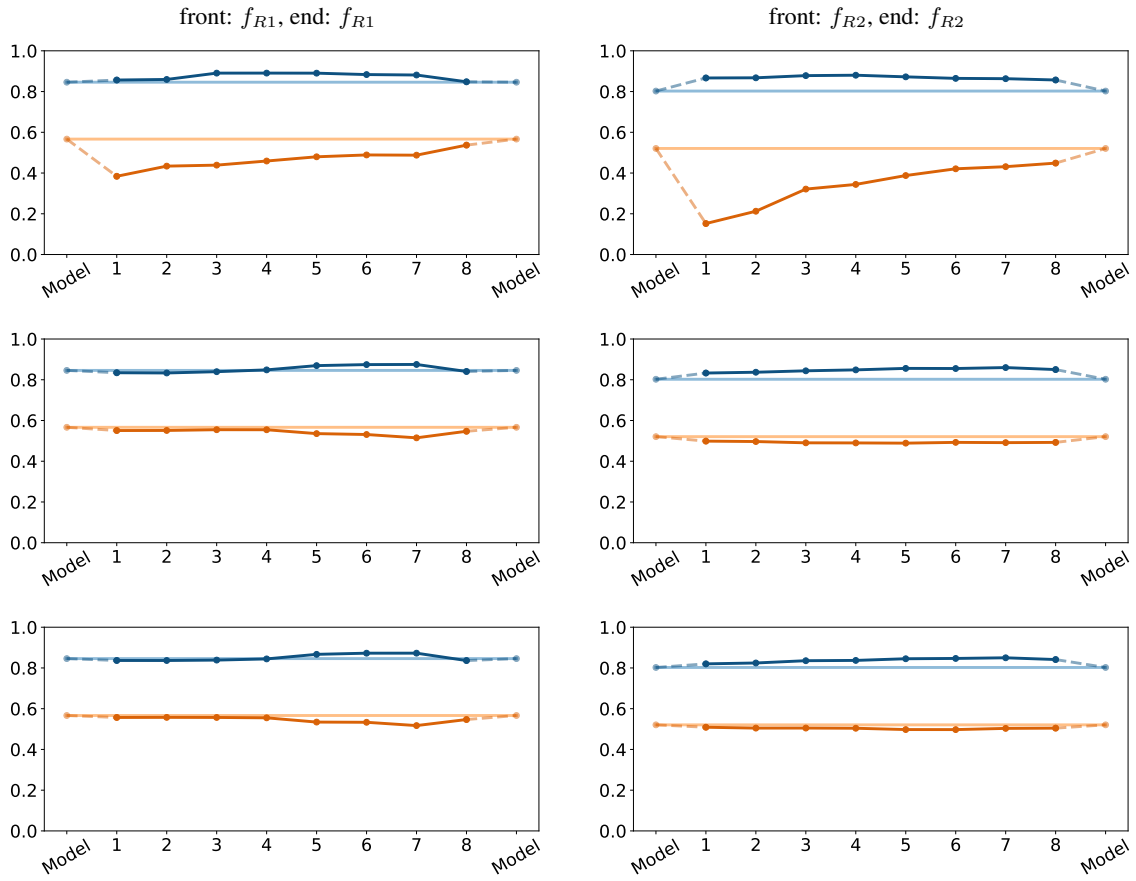

Figure 13. Stitching plots for identical front and end networks, with identity initialization, stitching optimized for plain accuracy (top row) and robust accuracy with $\alpha = 0.5$ (middle row) and $\alpha = 1$ (bottom row). (See Figures 2 and 3 for the same experiments but with random initialization.)

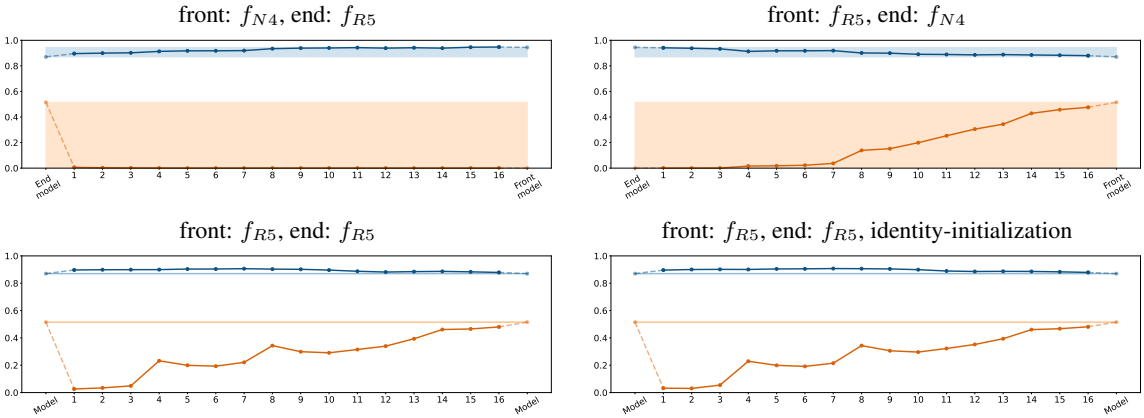

Figure 14. Results with the ResNet-50 networks, stitching optimized for plain accuracy.

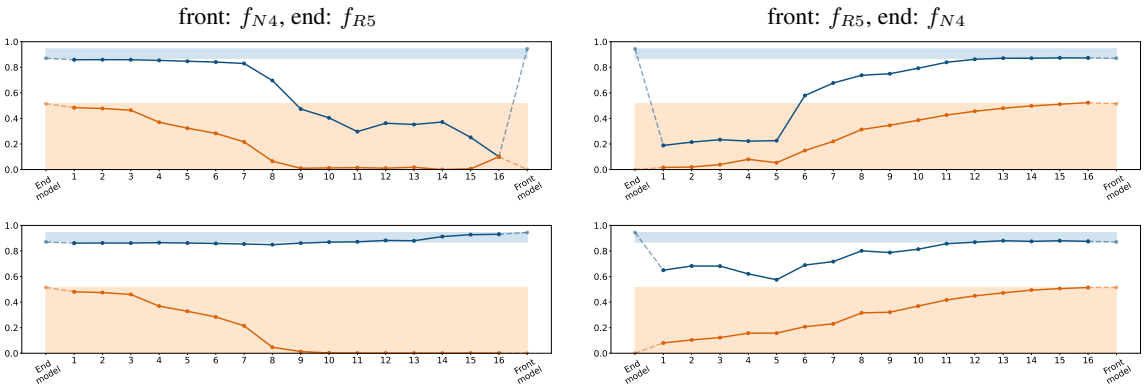

Figure 15. Results with the ResNet-50 networks, stitching optimized for robust accuracy with $\alpha = 1$ (top row) and $\alpha = 0.5$ (bottom row).

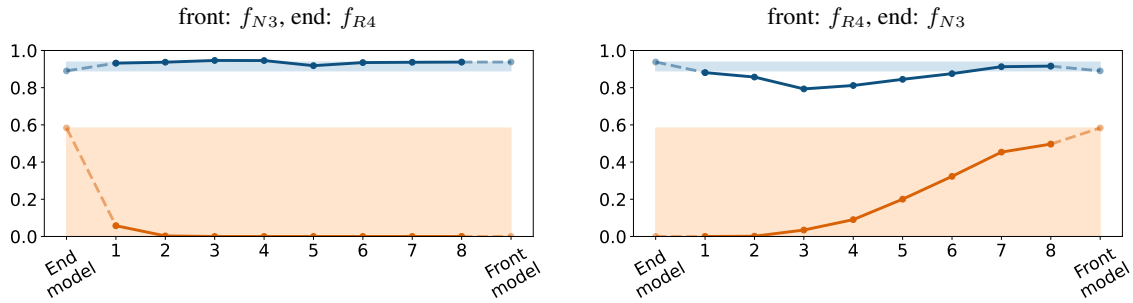

Figure 16. Results with the PreActResNet-18 networks, stitching optimized for plain accuracy.

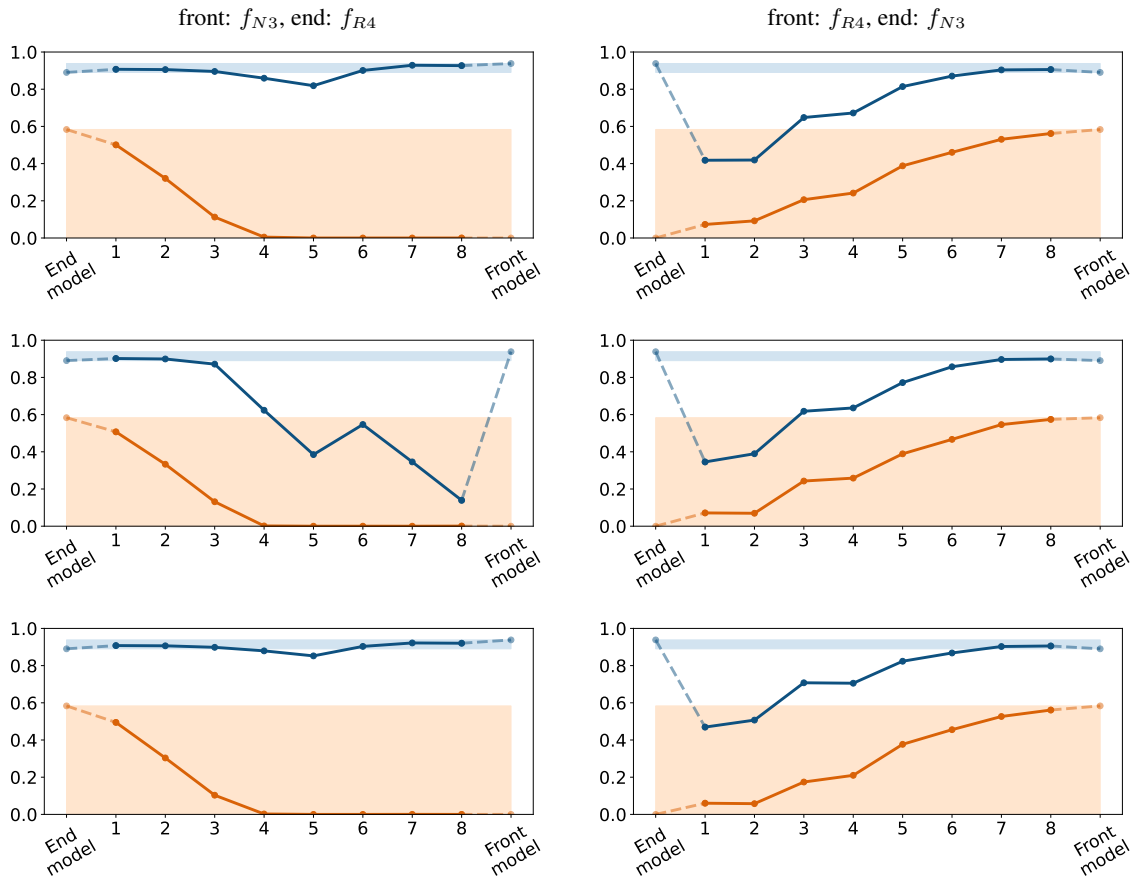

Figure 17. Results with the PreActResNet-18 networks, stitching optimized for robust accuracy with $\alpha = 0.5$ (top row), $\alpha = 1$ (middle row) and with TRADES, $\beta = 1$ (bottom row).

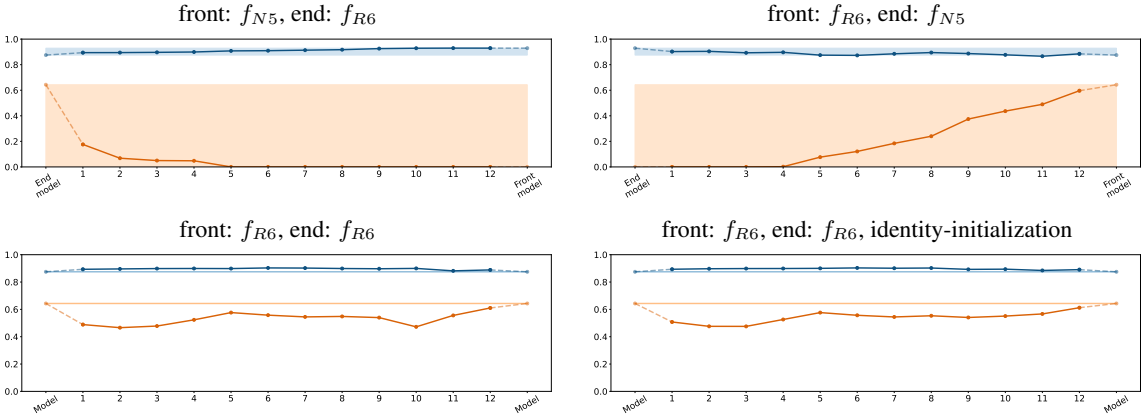

Figure 18. Results with the WideResNet-28-10 networks, stitching optimized for plain accuracy.

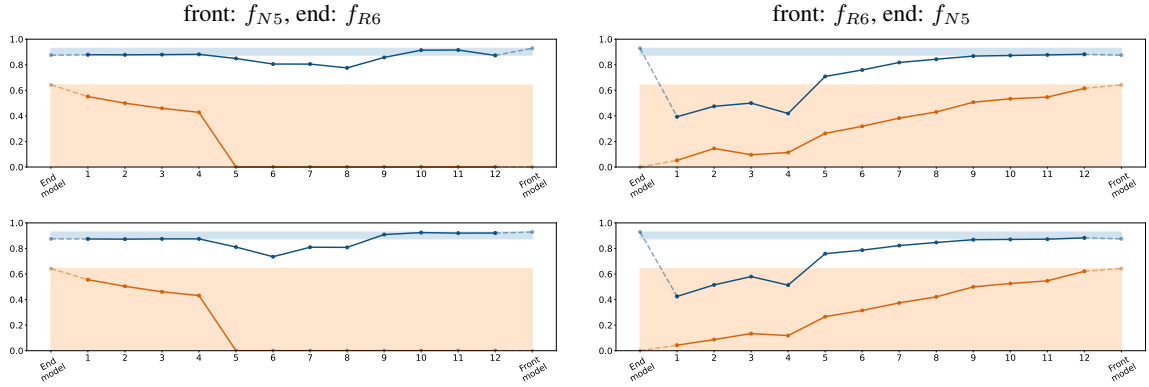

Figure 19. Results with the WideResNet-28-10 networks, stitching optimized for robust accuracy with $\alpha = 0.5$ (top row) and with TRADES, $\beta = 1$ (bottom row).

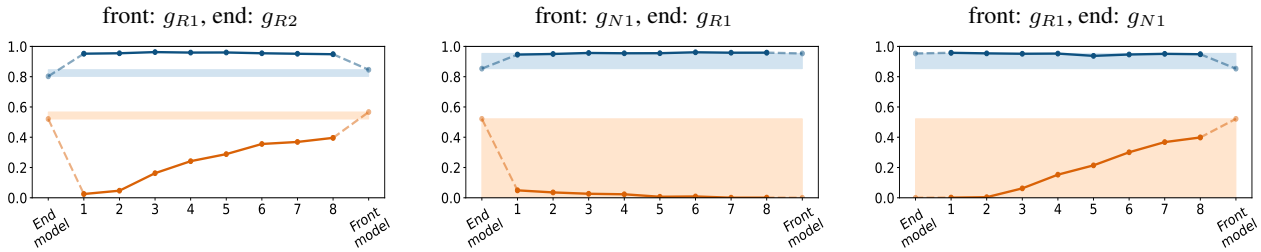

Figure 20. Results over the SVHN dataset, stitching optimized for plain accuracy.

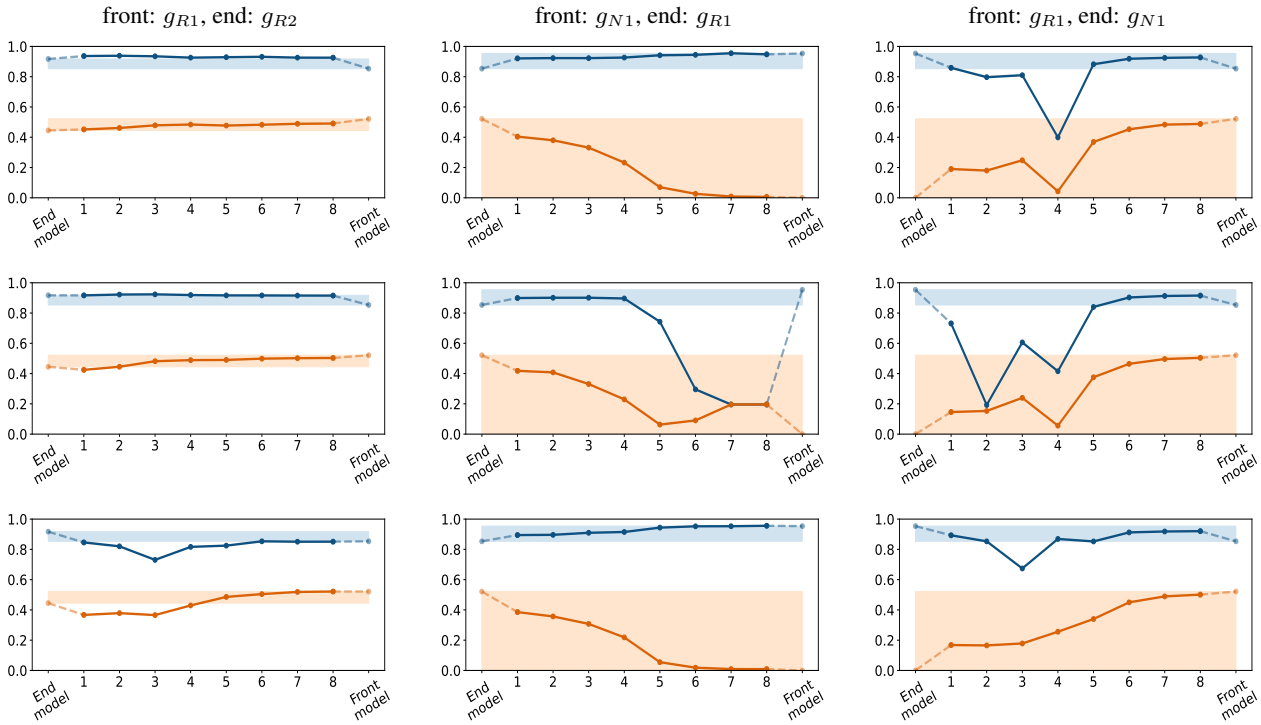

Figure 21. Results over the SVHN dataset, stitching optimized for robust accuracy with $\alpha = 0.5$ (top row), $\alpha = 1$ (middle row) and with TRADES, $\beta = 1$ (bottom row).

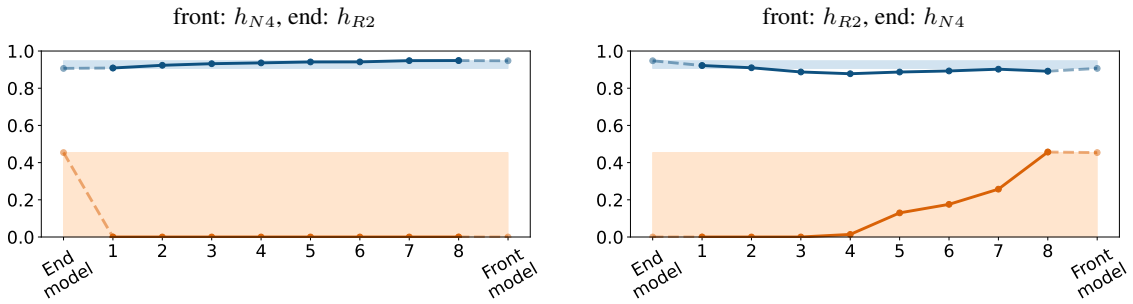

Figure 22. Results with Fashion-MNIST networks, stitching optimized for plain accuracy.

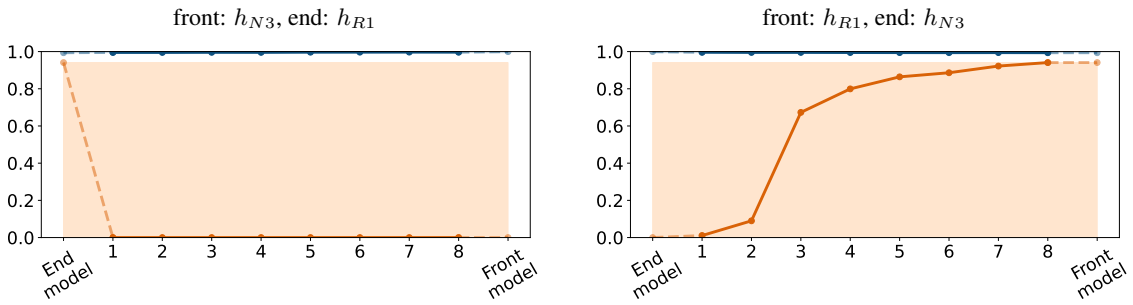

Figure 23. Results with MNIST networks, stitching optimized for plain accuracy.

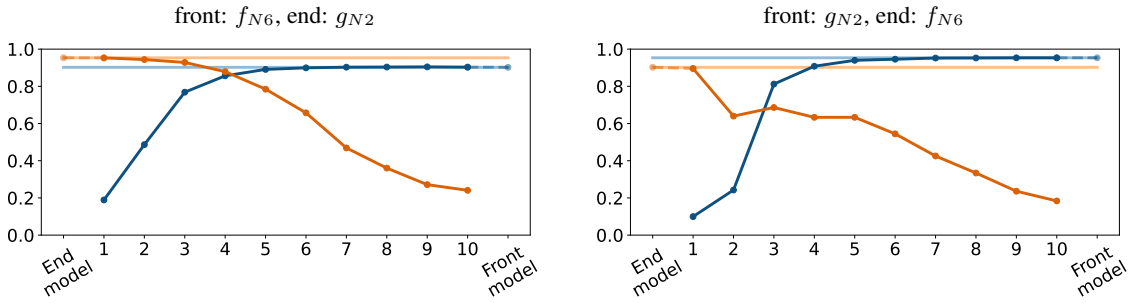

Figure 24. Cross-task stitching using CIFAR-10 network f_{N6} and SVHN network g_{N2} (both using VGG11 architecture). The curves show the plain accuracy of the stitched networks when stitching over CIFAR-10 and SVHN.

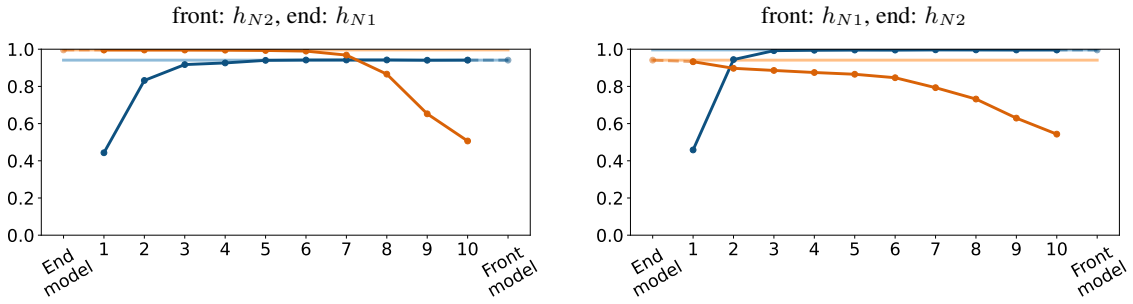

Figure 25. Cross-task stitching using MNIST network h_{N1} and Fashion-MNIST network h_{N2} (both using VGG11 architecture). The curves show the plain accuracy of the stitched networks when stitching over MNIST and Fashion-MNIST.

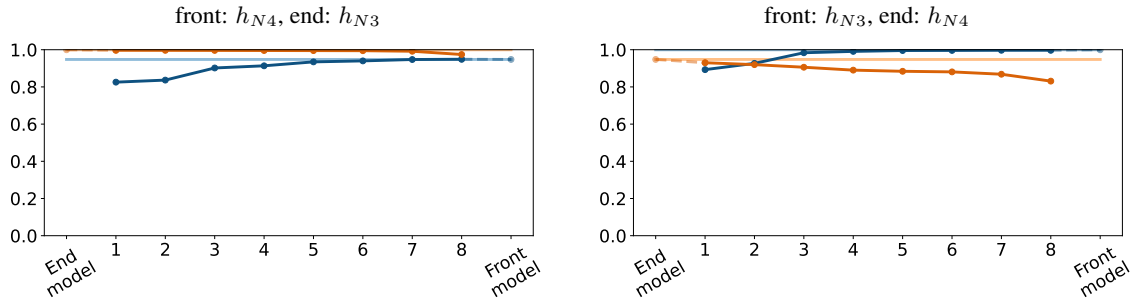

Figure 26. Cross-task stitching using MNIST network h_{N3} and Fashion-MNIST network h_{N3} (both using ResNet-18 architecture). The curves show the plain accuracy of the stitched networks when stitching over MNIST and Fashion-MNIST.

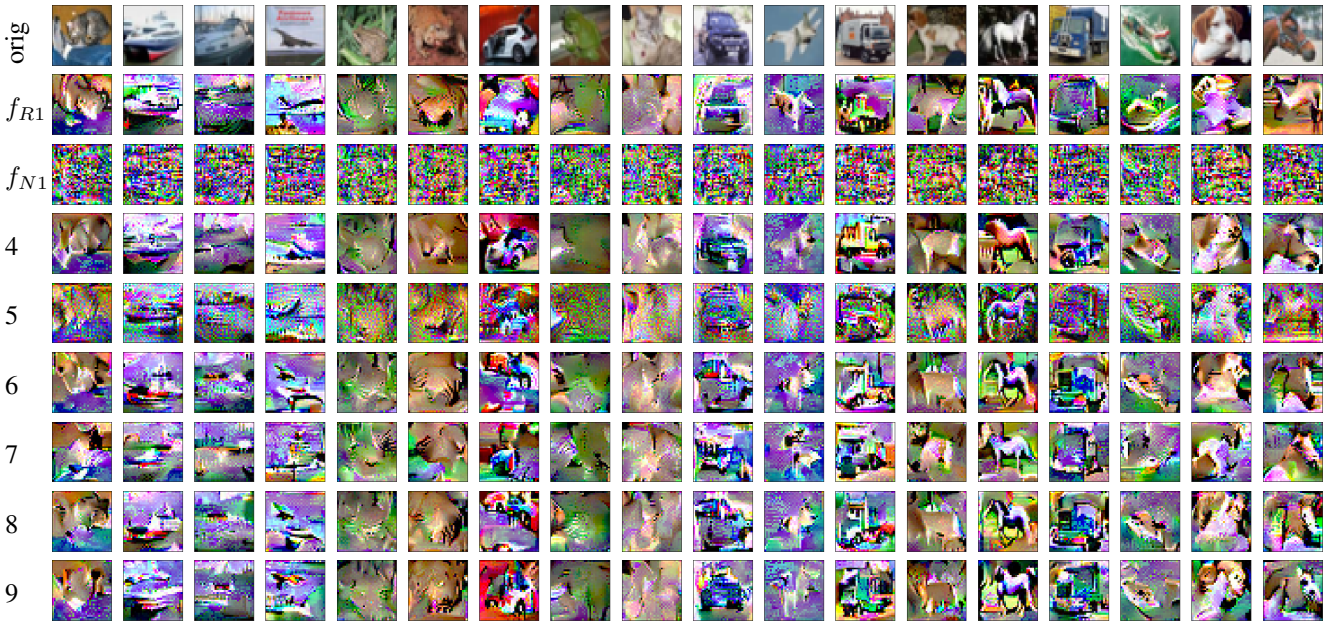

Figure 27. Inverted feature representations of several networks over a random sample of CIFAR-10 images (top row). In row 4 we have $f_{N1,>6} \circ T_{rob} \circ f_{R1,\leq 6}$, that is, the front network is f_{R1}, the end network is f_{N1}, robust stitching with $\alpha = 1$ at layer 6. Row 5 shows $f_{R1,>2} \circ T_{\text{rob}} \circ f_{N1,\leq 2}$ using robust stitching with $\alpha = 1$, Row 6: $f_{N1,>6} \circ T_{\text{plain}} \circ f_{R1,\leq 6}$, Row 7: $g_{N1,>6} \circ T_{\text{plain,cifar10}} \circ f_{R1,\leq 6}$ using plain cross-task stitching over CIFAR-10, Row 8: $g_{N1,>6} \circ T_{\text{rob,cifar10}} \circ f_{R1,\leq 6}$ using robust cross-task stitching over CIFAR-10 with $\alpha = 0.5$, Row 9: $g_{N1,>6} \circ T_{\text{rob,cifar10}} \circ f_{R1,\leq 6}$ using robust cross-task stitching over CIFAR-10 with $\alpha = 1$.