Comparing Apples and Oranges: is Stitching Similarity a Load of Spheres?

Damian Smith
School of Electronics and Computer Science
University of Southampton
ds5n23@soton.ac.uk

Antonia Marcu
School of Electronics and Computer Science
University of Southampton
a.marcu@soton.ac.uk

Abstract

Model stitching is used in the literature to assess the extent to which models capture similar information. The intuition is that if two models classify samples in the same way, they must be capturing the same information. We construct a series of experiments to show that two models can make the same predictions but represent very different information. We therefore argue that unlike previously claimed, stitching cannot reflect the extent to which models represent or capture similar information. This paper draws the community's attention to the need to correctly interpret the results of such functional similarity measures and highlights the need for similarity measures that capture informational similarity.

1 Motivation

Despite the success of Deep Learning, the community is still missing fundamental tools for analysing models. A useful such tool would be one that allows for model comparison. When trying to compare models, the *functional* perspective is gaining increasing attention [2, 7, 9]. This perspective argues that two models should be considered similar if they have **matching outputs**. Similarly, the argument is that two intermediate layers should be considered similar if they **lead to** matching outputs. While this perspective seems useful, especially in a classification setting, we argue that there is one fundamental problem with deep learning models which raises questions about its utility: shortcut learning [5].

Shortcut learning is an umbrella term for learning decision rules that rely on spurious correlations, causing the model not to perform well outside of the setting it was trained on. Geirhos et al. [5] make the point that many challenges in deep learning boil down to this issue. Moreover, the Simplicity Bias literature argues that SGD-trained models have a tendency for learning simpler rules [13, 4] (often shortcuts) over more complex rules that capture some of the true, salient, or intended information. Therefore, there is an increasing consensus that **models are prone to picking up shortcuts in the data**, whether we are aware of their existence or not.

In this paper we look at model comparison starting from the understanding that the models we compare have a propensity for learning shortcuts. We argue that in this context, **the functional perspective does not lead to a meaningful comparison.** In particular, we focus on Model Stitching [2], which is an increasingly popular method for comparing models' functional similarity. Informally, model stitching compares models A and B by "stitching" together the first part of model A with the last part of model B using a linear transformation referred to as a "stitch". The stitch is trained such that it learns a mapping between representations of models A and B. Model A is considered similar to model B if their stitched combination achieves a performance comparable to that of model B. In other words, model stitching assesses how compatible two models are from a functional perspective by comparing how the function represented by their stitched combination behaves compared to the B function when evaluated at the same input points. Functional compatibility is often taken to indicate that models capture similar information [e.g. 10, 2, 7].

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

In this paper we design experiments which demonstrate that the functional **compatibility** of models is inappropriate for evaluating the extent to which two networks capture similar information. Importantly, a model that learns a spurious correlation can appear to be fully compatible with one that learns the intended information. Given the prevalence of shortcut learning, we argue that a meaningful model comparison should distinguish between models that capture different information.

To do this, we first show that models which learn to use different information can easily be stitched together. We bias models towards learning to use either colour or shape information and achieve perfect stitching compatibility. We then consider a more challenging setting where we ensure that the sender not only does not use the same information as the receiver but its representations do not contain information relevant for the original receiver network. We do this by removing all shape information from the input data to be represented, leaving only colour information. We then stitch into a model that was trained on shape information alone and that is unable to make correct predictions based on colour information. We achieve full stitching compatibility in this case as well. Lastly, we show that we can successfully stitch clustered random noise into the models we train. We therefore show that we can easily construct cases where models' stitching compatibility does not reflect their informational similarity.

Our contributions are:

- We show that we can construct problems where stitching cannot distinguish between models known to have learned different shortcuts (i.e. learn to use different information) (Section 4.1);
- We show that representations which depict entirely different information or even clustered, random noise can be stitched into a trained model, raising further questions about the usefulness of stitching as a measure of model similarity (Section 4.2);
- Model stitching was also used to compare the "quality" of representations. We show that stitching provides conflicting evidence and we therefore argue it is inappropriate for comparing the quality of representations (Section 4.3).

2 Model stitching

Model stitching has seen many variants [e.g. 2, 3, 7] since it was first proposed [10]. For simplicity, in this paper we only consider stitching between identical architectures and at matching points in the network. Let $A_{\leq l}$ denote the composition of all layers in the trained network A up to and including layer l. Following Bansal et al. [2], we choose the stitching layer s to be a randomly initialised 1×1 convolutional layer preceded and followed by batch normalisation [8]. Bansal et al. [2] use a convolutional layer as it has restricted expressivity compared to a fully-connected layer. The untrained stitched model is therefore given by $B_{>l} \circ s \circ A_{\leq l}$. We then train the stitched model by freezing the parts taken from the models A (the sender) and B (the receiver) and only optimising the stitching layer s. We then report the performance of the stitched model on the test data. If the test performance of the stitched model is greater or equal to that of the receiver (which becomes the baseline), it is considered that models A and B are compatible at layer l.

In this paper we focus on ResNet-18 [6] models, but include results for VGG19 [12] in Appendix C.2. Note that we stitch ResNet-18 models before residual blocks and before the linear classifier. Therefore we refer to stitching before the first residual block as stitching at "Layer 1"; before the second residual block as "Layer 2"; and stitching before the classifier as "Linear".

Current stitching interpretation This work is inspired by Hernandez et al. [7] who find that stitching can reach high accuracy even when stitching from later layers in Model A to much earlier layers in a Model B. They believe that this could either be because the common intuition about how models process input is wrong, or because model stitching is able to match representations which are "different from what is expected". Nonetheless, they conclude that functional similarity provides a meaningful way of comparing models. Their intuition is that "if two representations can be used for similar purposes then in some sense they encode similar information" [7]. This agrees with Bansal et al. [2] who argue "two networks with identical architectures, but very different internal representations, would fail to be stitching connected".

We argue that no meaningful conclusion can be drawn from analysing the stitch connectivity. When stitching is **not** successful, this could simply be because a good enough mapping between the

representations was not found [3]. We claim that when stitching **is** successful, one cannot conclude that this is necessarily because the representations capture the same information. The next section shows the latter empirically.

3 Inducing a learning bias through the data

We choose to compare models on a typical shortcut learning problem: a variant of colour MNIST [1]. In essence, colour MNIST adds colour as an additional correlation between the input variable and the target variable. We chose to add colour as a fully-correlated background. For example, all instances of digit "0" have a red background, all instances of "1" have a green background, etc. We refer to this data set as "Correlated".

We use this problem to show that stitching is unable to distinguish between models that use different features to make the prediction. To do so, we want to simulate a scenario where **training a model leads to learning different shortcuts**. One possible way of biasing the model towards picking up different rules is to modify the architecture. However, modifying the architecture means that we can't perform a one-to-one model comparison. For this reason, we choose to modify the training data instead as a way of biasing the models. We therefore create different variants of the data set that lead to the model classifying based on either colour information alone, shape information alone, or different levels of relying on the combination between colour and shape.

Importantly, note that we modify the data in such a way that each model would still perform well (over 98% test accuracy) on the original Correlated data set. This means that the models could have been trained on the original Correlated data if we had an appropriate way of biasing them in a controlled way through the training procedure alone. Below we provide a brief description of the data set versions created (see Figure 1a for visual examples). For full details, see Appendix A. Source code is provided to assist in reproducing and extending this work at `https://git.soton.ac.uk/ds5n23/msc_similarity/-/tree/NeurIPSPaper?ref_type=heads`.

Colour MNIST (Correlated): The colour of the background and class of the digit are correlated.

Digit with Uncorrelated Colour (Digit): Images containing random combinations of background colour and digit. Target is given by the digit's class.

Colour with Uncorrelated Digits (Colour): Images containing random combinations of background colour and digit. Target is given by the colour's class. Importantly, the model could learn to represent shape information, but this cannot help the model classify.

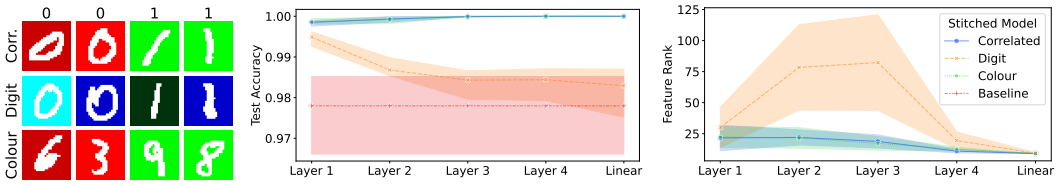
We refer to the models by the name of the data set they were trained on. Unless otherwise stated, the results are reported on ResNet-18 models. For full experimental details see Appendix B.

4 Experiments

4.1 Stitching cannot distinguish between different biases

In this section we want to verify if stitching can help distinguish between models that learn to classify using different information. We start by stitching various senders into the Digit model. Note that the stitching training and evaluation are performed on the Correlated data set, as this is the data we assume we have access to. All other data sets were created simply to train senders and receivers with different biases. We find that at all layers, all models we consider can easily be stitched into the Digit model, obtaining a higher accuracy than that of the original Digit model (see Figure 1b). That is, even the Colour with Uncorrelated Digits model, which learned to classify based on colour information (ignoring shape), appears compatible with the model that learned to classify based on shape information only.

Taking this further, we train a model on patches of colour with no digit ('Colour-only'). As opposed to Colour (which includes an uncorrelated digit), the model can only learn to represent solid colour information. Nonetheless, we obtain a stitching test accuracy higher than that of the receiver alone, indicating stitching compatibility (see Figure C5a).



(a) Example images (b) Accuracy (c) Rank

Figure 1: **(a)** Example images for classes 0 and 1 from each of the data sets. **(b)** Test accuracy on Correlated data when each of the trained models is stitched into the Digit receiver (baseline is the accuracy of the original Digit model on test Correlated data). At all layers, all models achieve higher average accuracy compared to the baseline, indicating compatibility. Shaded areas cover 100% of results to highlight the variability. **(c)** Rank analysis of the stitched models' representations. The representations are extracted from the receiver's first layer after the stitch. For layers 2 and 3, Colour and Correlated have a significantly different rank compared to Digit. Shaded areas mark 1 standard deviation.

Numerical rank We analyse the rank of the feature maps after the first layer of the receiving network. This is a way of gauging the linear dependence of the sender's feature maps as seen through the lens of the receiver's representation. Note that equal rank does not necessarily mean similar information, it just indicates a similar level of linear dependence. On the other hand, if two stitched senders lead to a different rank of the receiver, we take that as additional evidence that the senders have learned different information **according to the receiver**. If two sender representations have similar rank when processed by the receiver network, we cannot necessarily claim that they are perceived in a similar way. However, if they do not have the same rank, we take that as additional evidence that there exists a difference in how the receiver perceives them. Details of the numerical rank estimation are provided in Appendix B.2.1.

In Figure 1c we show the rank computed when stitching at each of the 5 layers we consider. We find that for stitching at layers 2 and 3 (before the 2^{nd} and 3^{rd} residual blocks) there is a clear gap between the rank of the processed representations of Digit with Uncorrelated Colour and those of Correlated, for example. This indicates that when fed into the receiving network, the representations of Digit with Uncorrelated Colour have a different degree of linear dependence compared to those of Correlated and therefore are not perceived as equivalent by the receiver despite both being stitch-compatible.

Arguably, colour information can still "leak" through a model that did not learn to use colour information for classification. Although less likely, the same argument holds for structured shape information leaking. We next perform two follow-up experiments: with a receiver model trained only on shape information, we obtain full stitch compatibility on Colour-only images (no digit is depicted), despite no shape information being present in the data; and stitching clustered, random noise into the receiving network. These experiments strengthen our findings that models can be easily stitched together even when they have very different internal representations.

4.2 Representations of very different information can be stitched together

To check whether stitching between dissimilar models is due to the sender model "leaking" information expected by the receiver, we remove all information that could be expected by the receiver from the images we train and evaluate the stitch on. Concretely, we stitch Colour-only models into receivers trained on Greyscale MNIST data using Colour-only data to train the stitch. There is no digit-like structure in the Colour-only data which (if leaked) could be used by the MNIST receiver. The un-stitched MNIST models have a baseline accuracy (when tested with MNIST data) of 0.990 ± 0.001; at every stitching layer, the Colour-only sender increases this to 1.0, resulting in stitching compatibility. I.e., despite the sender representations not containing any digit information at all as the input does not contain that information in the first place we are able to successfully stitch those representations into a Greyscale MNIST model.

Taking this further, we try stitching clustered, random noise into the receiver network. We create 10 random vectors to represent the mean of each of the 10 classes. The vectors have the same dimensionality as the representations that would normally be expected by the stitching layer. For

each class we create 6K samples by adding random noise to the mean class vector, obtaining a data set of the same size as the original Correlated data set (see Appendix A). We then stitch these representations onto the Digit receiver. We **can** achieve **full stitching compatibility** when using randomly generated representations. As expected, the rank of these representations when fed into the receiver network is different from than that of the learned representations (see Figure C5b). We found similar results with VGG19 models and provide the details in Appendix C.2.

In light of these results, we argue that models that capture significantly different information can, in at least some cases, be easily stitched together. While it may be argued that the datasets chosen are too simple, it is known that SGD-trained models find shortcuts [e.g. 13, 4], and there is not an algorithm to decide when stitching is applicable. Therefore, models' stitching compatibility should not be taken to mean that they learn similar rules or that they capture similar information.

4.3 Reverse stitching and stitching onto the same network

We then challenge the understanding that achieving higher stitching accuracy than the baseline (receiver's own accuracy) means the representation of the sender network is "better" than that of the receiver. To this end, we simply swap the receivers and the senders considered in the previous experiments. In Figure 1b we observed a higher test accuracy for Colour and Correlated compared to the Digit baseline, which would be taken to indicate that their representations are "better" for discriminating samples on the problem we consider. However, stitching Digit (as sender) into Colour or Correlated (as receivers) also leads to higher accuracy for particular layers compared to the baseline (see Table C1), which is a contradiction. Therefore, we do not think stitching accuracy is suitable for comparing the quality of models' representations. Finally, we stitch a model with itself (Digit vs Baseline) and obtain an increase in accuracy (see Figure C5a) as well as a change in the rank of representations (see Figure C5b). This further indicates that although the stitching layer has reduced expressivity compared to a fully-connected stitch, it still cannot be claimed that it simply aligns the representations of the sender and receiver without any additional processing.

5 Conclusions

In this paper we showed that networks can use or represent very different information, yet classify samples with similar accuracy. Importantly, the different representations can easily be stitched together. This leads us to question the usefulness of studying models' functional similarity, and in particular their stitching compatibility, to determine whether or not they capture similar information. We hope that our work encourages the community to more carefully interpret the results of model stitching, in particular to understand what it is actually responding to. We also hope our work will encourage researchers to focus on creating model comparison tools that can reliably capture informational similarity. We propose artificial shortcut learning problems as a good starting point for starting to reason about this in a controlled way.

6 Acknowledgements

We would like to thank the University of Southampton School of Electronics and Computer Science for their support of this work.

References

[1] H. Bahng, S. Chun, S. Yun, J. Choo, and S. J. Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning (ICML)*, 2020.

[2] Y. Bansal, P. Nakkiran, and B. Barak. Revisiting Model Stitching to Compare Neural Representations, June 2021. URL `https://arxiv.org/abs/2106.07682v1`.

[3] A. Csiszárik, P. Kőrösi-Szabó, A. Matszangosz, G. Papp, and D. Varga. Similarity and matching of neural network representations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5656–5668. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/2cb274e6ce940f47beb8011d8ecb1462-Paper.pdf`.

[4] G. De Palma, B. T. Kiani, and S. Lloyd. Deep neural networks are biased towards simple functions. *arXiv*, 2018, 2018.

[5] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[7] A. Hernandez, R. Dangovski, P. Y. Lu, and M. Soljacic. Model stitching: Looking for functional similarity between representations. In *SVRHM 2022 Workshop @ NeurIPS*, 2022. URL `https://openreview.net/forum?id=7bHLCO5FQdB`.

[8] S. Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[9] M. Klabunde, T. Schumacher, M. Strohmaier, and F. Lemmerich. Similarity of Neural Network Models: A Survey of Functional and Representational Measures, Aug. 2023. URL `http://arxiv.org/abs/2305.06329`. arXiv:2305.06329 [cs].

[10] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[11] W. Masarczyk, M. Ostaszewski, E. Imani, R. Pascanu, P. Miłoś, and T. Trzciński. The Tunnel Effect: Building Data Representations in Deep Neural Networks, Oct. 2023. URL `http://arxiv.org/abs/2305.19753`. arXiv:2305.19753 [cs].

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[13] G. Valle-Perez, C. Q. Camargo, and A. A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

Figure A1: Colour swatches for the base colour_map

A Data sets

Training used variants of MNIST data

- handwritten digits, label is digit represented
- greyscale
- 28x28 pixels
- trainset: 60 000 images
- testset: 10 000 images

The following variants were created:

MNIST This is simply MNIST data (monochrome handwritten digits) expanded to 3 channels and normalised.

There is no target-dependent colour information for the model to learn, so we *expect* it to recognise the digits based on features such as shape, or the amount of white, or the texture of the white-grey-black edges. It *cannot* learn to use colours to recognise classes as these are never present.

It is possible that kernel weights will learn to rely on the three colour channels always being equal.

Colour MNIST (Correlated) We wanted to provide an opportunity for some semantically different representations to be learned. In this case, having the background colour correlate with the target digit being displayed.

We *expect* that the models trained on Correlated data will mostly learn to rely on the colour. However, they *may* learn the shapes of the digits, or the amount of white caused by the digit. They may rely on pixels always being at one of two rgb values (white/colour) in any one image, never at an intermediate value. They may learn to rely entirely on a small number of pixels that are never white - e.g. corner pixels.

To generate the 'bias' dataset, we used the *colour_MNIST* package [1] which uses MNIST data, but changes the colour of the background depending on the digit being represented (Figure A1). The package snaps any non-zero pixels to white (`[255,255,255]`). This could be thought of as changing the greyscale information to binary, and indeed reducing the information content as a result. The background colour is changed from black (`[0,0,0]`) to one of 10 values, always matching the data label.

For all of these datasets, to avoid the model learning something as simple as specific RGB values, the base RGB background values are modified by a random 10% per image (such that the colour is always flat, but will vary slightly from image to image within a class). The same variation is injected into the test datasets. This is analagous to requiring that colour is learned in a generalised way.

Digit with Uncorrelated Colour (Digit) To encourage models to learn representations which recognise digits, but cope with different background colours, we created a dataset with digits over laid on backgrounds whose base colours were randomly selected. i.e. the digit does *not* correlate with the colour.

We expect models trained on Digit to learn features like shape. Assuming the randomisation is sufficient, they *cannot* learn to rely on colour. The choice of base colours and inclusion of colour variation mean that the Digit models *cannot* learn to rely on a single colour channel or pair of colour channels, and *must* learn to tolerate a range of colours being used. Nonetheless, they *may* learn to rely on any or all pixels in an image being one of two colours.

Figure A2: Simulated "Clustered Noise" data sample for ResNet-18 at Layer 2, showing 64 7x7 feature maps. If this represented Class 0, all other Class 0 data instances would be noisy versions of this.

Colour with Uncorrelated Digits (Colour) To encourage representations of colour information, but which are able to tolerate the presence of digits, we generate a dataset in the same way as for Digit but in which the background colour (not the uncorrelated digit) is the target.

Models trained on Colour *cannot* use the digits to classify because they are uncorrelated with the background colours. Successful Colour models *may* rely on colour being sampled in specific locations which never contain digit pixels (e.g. a corner), or might learn to average across the image. They may learn to rely on the presence of some white pixels as the dataset does not contain any solid-colour images. But they may not be generally immune to non-digit-like patterns of white pixels.

Background Colour Only (Colour-only) To force the learning of different representations which *cannot* be related to shape, size, or edge-effects of digits (and which may not be able to tolerate those features), the 'Colour-only' dataset was created in which no digit image is present, only a solid background colour (plus 10% variation). As in 'Colour' (see Figure 1a), the base colour is the target.

The model *may* learn to rely on the colour in only one region of the image, or to assume the colour is constant everywhere.

Clustered Noise (Noise) To produce a high-quality representation, without relying on specific, image-related features, we generate synthetic datasets in the form of clustered noise.

Passing an image through the first layers of a sender model produces a set of activations specific to the final layer before the cut. For example, with ResNet-18 cutting just before Layer 2 of the receiver, the sender output will be 64 channels of size 7x7 (Figure A2.) The synthetic dataset must be matched in shape to the layer it is being stitched at. As such, it has to be regenerated depending on the layer in question. It does not produce images, rather synthetic activations. Each data instance is represented by a point in activation space based on its class, plus a random offset vector. Thus each class is located in a cloud or ϵ-ball. For example, in the Layer 2 example, each of the ten classes will be centred around a different random point in 3136-dimensions.

There is no attempt to create structure within or between the feature maps, and it is not derived from actual activation data. The high-dimensionality means it is likely that the clusters of data points will be highly separable, though this is not enforced and has not been verified.

To ensure the same base representations are used for train and test datasets for a given layer or test, a `genearate_activations()` function was created (Alg 1). A train and test dataset can then be generated (Alg 2) and accessed via a dataloader.

Algorithm 1 Generate Base Activations

Procedure GENERATE_ACTIVATIONS($num\_classes$, $representation\_shape$)
for $c \leftarrow 0$ **to** $num\_classes - 1$ **do**
  $activations[c] \leftarrow rand(representation\_shape)$
end for
return $activations$
End Procedure

Algorithm 2 Generate a dataset (unshuffled)

Procedure SYNTHETICDATASET($train$, $activations$, $noise$)
if $train$ **then**
  $SamplesPerClass \leftarrow 6000$
else
  $SamplesPerClass \leftarrow 1000$
end if
for $c \leftarrow 0$ **to** $num\_classes - 1$ **do**
  $data[c * SamplesPerClass : ((c+1) * SamplesPerClass)] \leftarrow activations[c]$
  $targets[c * SamplesPerClass : ((c+1) * SamplesPerClass)] \leftarrow c$
end for
$data \leftarrow data + noise * randn$
$data \leftarrow clamp(data, 0, 1)$
End Procedure

B Experimental details

B.1 Stitching between models

In an extension of the experiments by [2], we train models on different special datasets (Section A) and then stitch between them, assessing the change in accuracy.

Hyperparameters for Model Training batch_size=128, 4 Epochs, SGD, lr=1e-1, momentum=0.9, weight_decay=1e-4.

Hyperparameters for Model Stitching batch_size=128, 10 Epochs, SGD, lr=1e-4, momentum=0.9, weight_decay=1e-2

Hyperparameters for Noise Stitching batch_size=64, 4 Epochs, SGD, lr=1e-4, momentum=0.9, weight_decay=1e-2

Two versions of this are performed, one stitching from each different model into the Digit with Uncorrelated Colour receiver model, and the other stitching from that model as sender into each of the others. This will allow us to assess the symmetry of the stitching process. Recall that models trained on each of the datasets represents a model that *could* have arisen naturally by training on Correlated data, but with the advantage that we know something about features that may (or cannot) be learned.

In this experiment, all image datasets are used to train models. Each of the models (including Digit) is then stitched into the Digit model at each of 5 points: (e.g. see Figure B3).

- Cut before ResNet Layer 1
- Cut before ResNet Layer 2
- Cut before ResNet Layer 3
- Cut before ResNet Layer 4
- Cut before Linear Layer

The accuracy and rank are then measured using the Correlated test dataset. Accuracy and rank are also measured for the whole, unstitched 'Digit' model using the Correlated test dataset for comparison. This test configuration is different from that used by [2] and [7]. They used the same dataset for training and testing their models. We wanted to examine the case where:

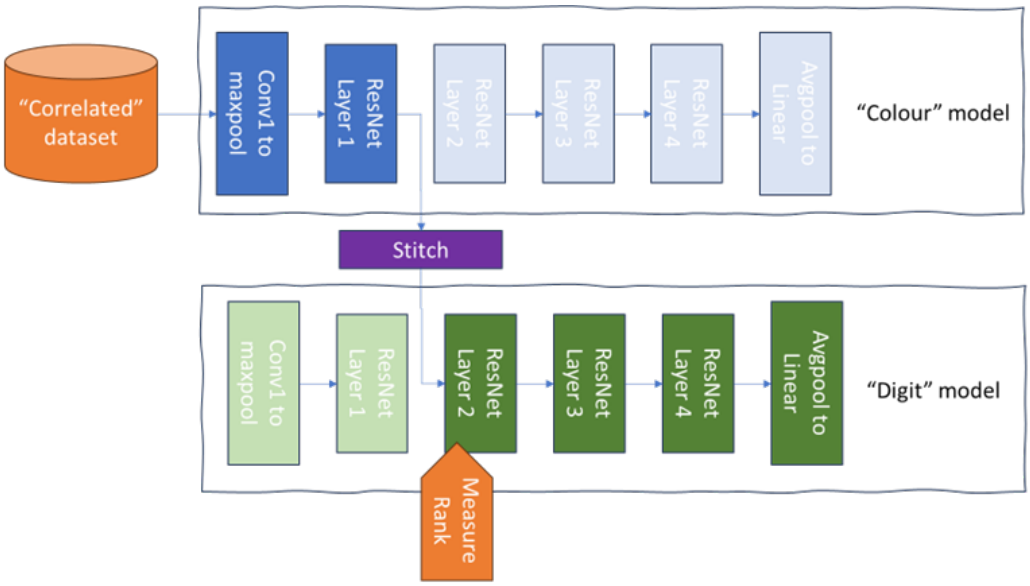


Figure B3: An example model to model stitch test setup. Correlated dataset is always used. Digit model is always the receiver. In this example, the Colour model is the sender and the cut is before ResNet Layer 2. Note that the Rank is measured at the first conv layer of the Digit receiver model

1. The dataset for training and testing the stitch is correlated (biased) - i.e. it contains class-correlated features other than the intended learning target (the written digit): in general this will be the case, even if unintentionally. Recall that the bias is that the background colour is correlated with the labelled digit.
2. The sender models are *likely* to have learned different features from the receiver.

B.2 Stitching from noise

We train ResNet-18 models for each of the image-based datasets described. For each model, the accuracy is recorded for the associated test dataset. The model is cut before the first ResNet block, and prepended with the stitch to create a stitch + receiver model. The stitch (only) is then trained on the noise dataset.

The stitch + receiver model (Figure B4) is then tested against the synthetic test dataset. This is repeated, cutting before each ResNet layer (a layer comprises 2 Basic Blocks. The skip connections are preserved), and before the final fully-connected linear layer. We also record the rank of the activations for the Noise test dataset at the first convolutional layer in the receiver model after the stitch. This is to provide insight into how the stitched data is "perceived" by the different models. Also how that compares to when dataset images are processed by the model. For each test at each stitch point the synthetic dataset was regenerated to reduce the likelihood that results are due to one randomly selected set of points having a significant structure.

B.2.1 Examining the rank

To gain further insight into how the representations being sent are "perceived" by the receiver, we analyse the rank at the first receiver layer. Note, however, that generally analysing the rank of two sets of representations cannot tell us anything meaningful about their informational similarity. We are simply arguing that the receiver cannot map the various sender representations in an entirely equivalent way if the rank in the first receiving layer is different.

1. Train several models each on a different, but related dataset. For example, Colour MNIST can create different dataloaders of MNIST digits on coloured backgrounds where the colours correlate or do not correlate with the digit class.

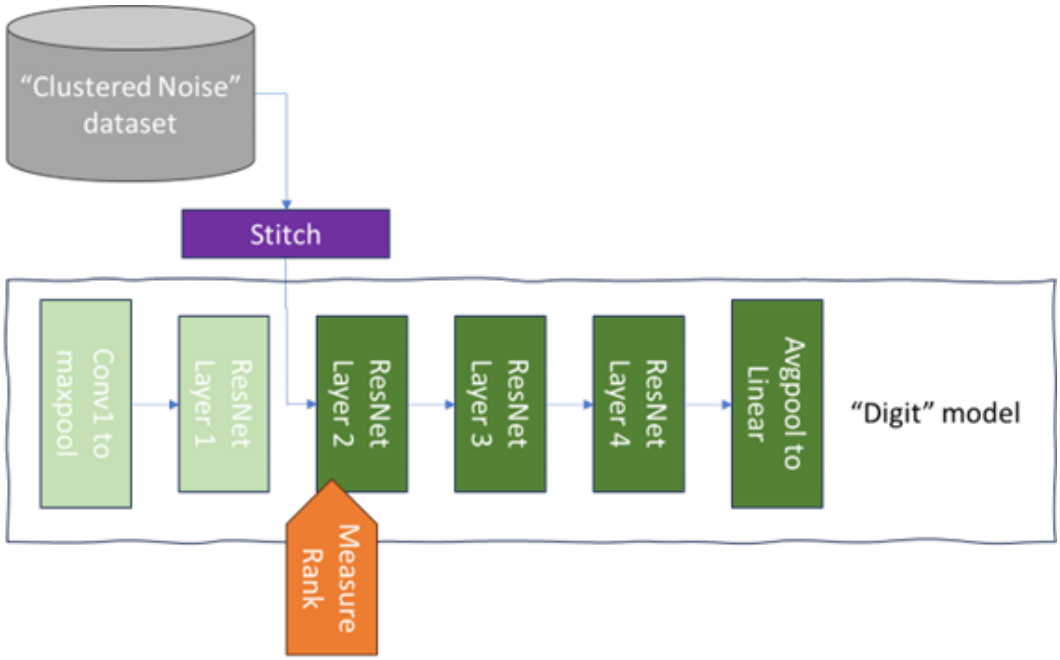


Figure B4: In this example, the Digit model is the receiver and the cut is before ResNet Layer 2. Note that the Rank is measured at the first conv layer of the Digit receiver model. The shape of the 'Noise' data must match the receiving layer.

Table C1: Accuracy of trained ResNet-18 models against 'Correlated' dataset (average of 5 initialisations).

		Acc. with stitch at Layer 1	
Model	Base Accuracy	'Digit' is Receiver	'Digit' is Sender
Correlated	0.999	0.999	0.981
Digit	**0.978**	0.995	0.996
Colour	1.000	0.999	1.000
Colour-only	**0.686**	**0.999**	**0.985**

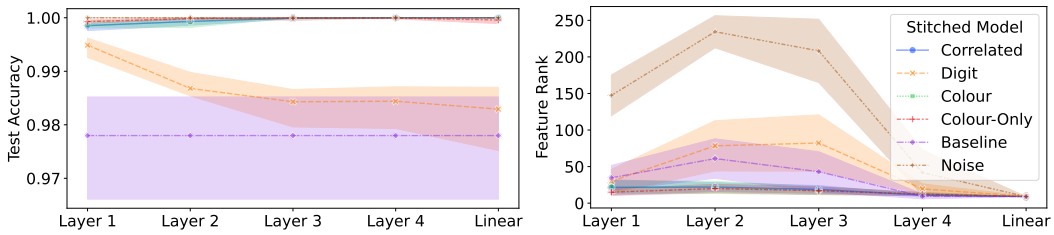
2. Stitch models to each other such that sender is from layer 1 to L, and receiver is from layer L+1 to Classifier. Train the stitch.
3. Collect the representations at layer L+1 obtained using the test dataset. For estimating the rank, we follow [11], who compute the singular values of the sample covariance matrix and threshold these at 1e-3 of the largest singular value.

C Additional results

C.1 All ResNet-18 models stitched to Digit with Uncorrelated Colour

As mentioned in Section 4.3 and Appendix B we used the Digit model as both receiver and sender in the stitch, and also used clustered noise as a synthetic sender model. Here we present those extended results.

In Table C1 we can see that the base accuracy of the Digit network is 0.978. When stitching Colour-only into this, accuracy improves to 0.999, indicating that Colour-only is "better than" Digit. However, the base accuracy (against the same test set of Correlated) of the Colour-only network is 0.686, but when stitching Digit into it as a sender, the accuracy improves to 0.985, indicating that Digit is "better than" Colour-only. This contradiction constitutes a problem for the claim that model stitching can be used to ascertain relative quality. Note also that stitching the Digit model into itself produces a



(a) Accuracy (b) Rank

Figure C5: **(a)** Test accuracy on Correlated data when the 'Noise', and each of the trained ResNet-18 models are stitched into the Digit receiver (baseline is the accuracy of the original Digit model on test Correlated data). This extends Figure 1b by including Colour-only and Clustered-Noise. Shaded areas cover 100% of results to highlight the variability. **(b)** Rank analysis of the stitched models' representations. Shaded areas mark 1 standard deviation.

Table C2: Accuracy of trained models against same test dataset (average of 3 initialisations). Values shown to aid reading of Figure C6a

Model	Test Accuracy
'Correlated'	1.00
'Digit'	0.98
'Colour'	1.00
'Colour-only'	1.00

performance improvement from 0.978 to 0.995 which challenges the belief that the 1×1 convolution with batchnorms stitch [2] does not add capacity.

C.2 VGG19 results

There is a reasonable question about whether ResNet architectures may respond to model stitching in a particular way because of the skip connections. To address this, we extended the testing with clustered noise data to include VGG19 models. 3 initialisations were run. 50 Epochs for model and stitch training: batch_size=64, SGD, lr=1e-2, momentum=0.9, weight_decay=1e-4.

Each trained model was cut and stitched at the following points. We decided to take a sample rather than testing every convolutional layer to reduce experimental duration (Figure C6a):

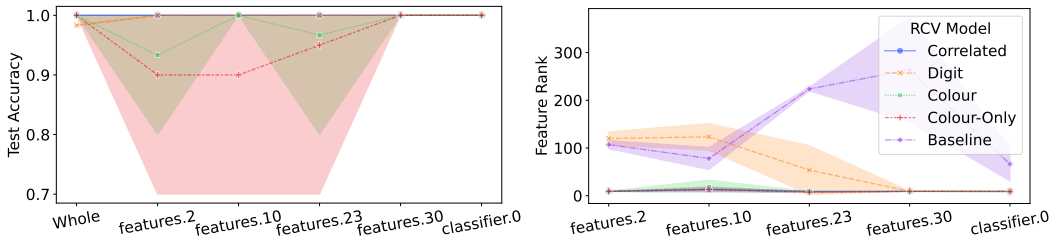
- Whole Model (image data)
- Cut before features.2 (Noise data)
- Cut before features.10
- Cut before features.23
- Cut before features.30
- Cut before classifier.0

We record the rank of the activations for the synthetic test dataset at the first convolutional layer in the receiver model after the stitch (Figure C6b).

Figure C6a and Table C2 show that 'Digit' has improved results with stitching in the 'Noise' dataset. 'Correlated' maintains accuracy. This is in accordance with the results for ResNet-18.

Most notably, 'Colour' and 'Colour-only' show very varied results with stitching. It is not clear from this limited experiment whether that is due to the network not being stitching-compatible with the synthetic data, or if it is a manifestation of the random variability shown by [3] (i.e. stitching from different stitch initialisations can yield very different stitching-penalties.)

We carried out a small investigation using the model which performed worst at features.2 stitching. Trying multiple stitch initialisations gave accuracies in the range $\sim 40\% - 80\%$. Trying multiple



(a) Accuracy (b) Rank

Figure C6: **(a)** Accuracy of VGG19 models trained on the four dataset variants when cut with a stitch at 5 different points in the model. The first sample shown ('Whole') is for the un-cut models tested with their own test datasets. Shaded areas cover 100% of results to highlight the variability. **(b)** Ranks of activations just after the stitch from synthetic Clustered-Noise test dataset. 'Baseline' shows ranks at the same points of the uncut 'Digit' trained VGG19 models when presented with 'Digit' test data. Shaded areas mark 1 standard deviation.

Noise dataset initialisations gave similar results. Lower and Higher learning rates were also tried with no change. It may be that the training parameters are wrong as sometimes a good stitch-training loss increases between epochs, and we did not optimise stitch training hyperparameters. However, it may be that some models reach points in the loss landscape which it is hard to match when stitching from 'Noise' data.

This may represent a difference between ResNet-18 and VGG19 architectures. Two possible reasons (which deserve to be investigated) are:

skip connections ResNet skip connections may make stitching easier.

dimensionality VGG19 at features.2 has 64 feature maps of size 32x32, whereas Layer 2 of ResNet-18 has 64 feature maps of size 7x7. The 1x1 Convolutional stitch layer can only learn to create linear combinations of its input feature maps and it may be harder to create necessary patterns of activation when (in VGG19) the maps have ~ 21 times as many units.

The most obvious feature of the rank information in Figure C6b is the difference between ranks in the unstitched 'Digit' network (presented with 'Digit' data) and the stitched networks presented with 'Noise' data. This demonstrates that being stitched to a sender can produced similar functional results in terms of overall accuracy, while internal information is different.

D Source code

For carrying out rank calculations the following repository (branch "NeurIPSPaper") should be placed next to the msc_similarity repository `https://github.com/DHLSmith/jons-tunnel-effect/tree/NeurIPSPaper`