ON A BUILT-IN CONFLICT BETWEEN DEEP LEARNING AND SYSTEMATIC GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Out-of-distribution or systematic generalization is a desirable property that most deep learning algorithms lack. In this paper, we hypothesize that internal function sharing is one of the reasons to weaken systematic generalization in deep learning for classification tasks. Under equivalent prediction, a model partitions an input space into multiple parts separated by boundaries. The function sharing prefers to reuse boundaries, leading to fewer parts for new outputs, which conflicts with systematic generalization. We show such phenomena in standard deep learning models, such as fully connected, convolutional, residual networks, LSTMs, and (Vision) Transformers. We hope this study provides novel insights and forms a basis for new research directions to improve systematic generalization. Source codes are available in the supplementary material.

1 INTRODUCTION

A fundamental property of artificial intelligence is generalization, where a trained model appropriately processes unseen test samples. Many problems adopt the i.i.d. assumption. On the other hand, out-of-distribution (o.o.d.) or *systematic generalization* (Fodor & Pylyshyn, 1988; Lake & Baroni, 2018) requires that training and test distributions are disjoint so that test samples have zero probability in training distribution. Systematic generalization is crucial for human learning and supports efficient data use and creativity. Therefore, machines are encouraged to acquire such generalization ability to achieve human-like intelligence.

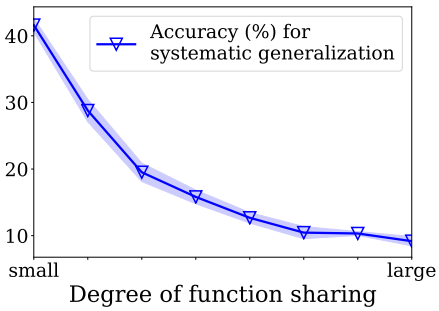
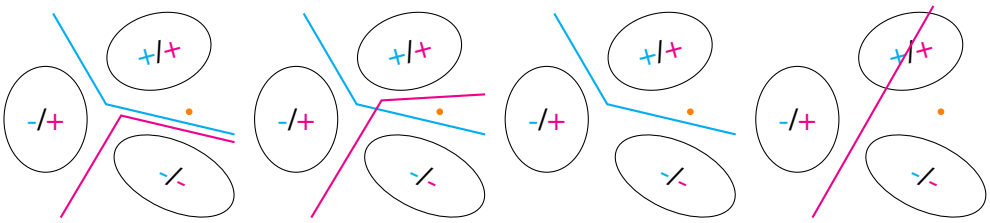


Figure 1: A simplified plot for a result in the experiment section. As the degree of function sharing increases (more sharing layers), the accuracy of the test dataset or generalization capacity decreases accordingly. It indicates that function sharing in deep learning conflicts with systematic generalization. Please refer to Section 3 and Figure 5a for details.

Systematic generalization usually requires that a sample has multiple explanatory factors of variation (Bengio et al., 2013), and the generalization is enabled by producing an unseen combination of seen factor values. For example, models trained on blue rectangles and green triangles predict blue triangles. We adopt factors mainly in designing experiments and developing intuitions. It helps experiments because new outputs are only related to function sharing between factors (Section 3). So we limit our claim to the cases for recombination of factors.

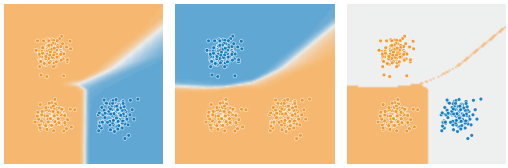
One stream of artificial intelligence is Connectionism (Feldman & Ballard, 1982; Rumelhart et al., 1986), which uses many simple neuron-like units richly interconnected and processed in parallel. It was criticized that Connectionist models do not support systematic generalization well (Fodor & Pylyshyn, 1988; Marcus, 1998). *Deep learning* (LeCun et al., 2015) originates from Connectionism, and various techniques have enabled multiple-layer modelings and improved performance on i.i.d. problems in recent years. Also, specific algorithms have been proposed to equip deep learning with systematic generalization ability (Russin et al., 2019; Lake, 2019). However, less discussion has been made on why standard deep learning models do not achieve systematic generalization.



(a) DL preferred model (b) DL unpreferred model (c) First function (d) Second function

Figure 2: Intuitions for the model preference of deep learning (DL). Each figure is an input space containing three sets of training samples with outputs of +/+, -/-, and -/+, respectively. Models have two decision boundaries (cyan and magenta). The orange dot is a test sample with a new ground-truth factor combination +/-. We discuss that the training process prefers the first case (a) over the second one (b) because it likes to share or reuse functions. Suppose the first function (c) is learned, then the process tends to reuse it by learning a simple function (d) and combining them instead of learning the complicated magenta function in (b) from scratch. So the function between +/+ and -/- is shared. Few inputs are mapped to the new output, and systematic generalization is not achieved.

This paper addresses the above question by looking into a built-in characteristic of deep learning. A node (or a set of nodes) is a function of the network input. By *function sharing*, we mean nodes in one layer share inputs from nodes in the previous layer. We hypothesize that function sharing is one of the reasons to prevent systematic generalization in deep learning. Under equivalent prediction, a classification network partitions an input space into multiple parts separated by boundaries. Function sharing prefers to reuse boundaries and avoid redundant ones for training predictions. It leads to fewer parts for new outputs and weakens systematic generalization. The nearest neighbor classifier is an analogy for this effect because it predicts a training sample output for any test sample, so it does not have a part for new outputs. We also discuss that function sharing belongs naturally to deep learning (Section 4).



First output Second output Result

Figure 3: A visualized example with a two-dimensional input and two binary outputs. The model is a multiple-layer fully connected neural network. The three figures contain the predictions for the first and the second outputs and classification results for the model. Each figure is the input space. The dots are training samples whose colors are ground-truth labels. For output figures, the background colors are model predictions. We set the new combination of labels (blue-blue) as the correct pair of outputs. For the result figure, the background is orange if both predictions are wrong, white if one is correct, and blue if both are correct. The result figure does not have a blue background, indicating that the model does not predict the new label combination for any input.

Figure 2 has an intuitive example explaining why the conflict happens. A test sample (in orange) equals a set of training samples (+/+) on the first output. Then they are also equal on the second output if the function is reused (see caption). Therefore, they are equal in all the outputs, which conflicts with systematic generalization. More generally, the two boundaries jointly partition an input space into multiple parts, and deep learning prefers (a) because it has fewer parts than (b). Figure 3 has a visualized example. It is similar to the example in Figure 2a. The two functions are shared in the top-right region, and few inputs are predicted as the new combination. Figure 1 has a simplified plot for a result in the experiment section. As the degree of function sharing increases (more sharing layers), the accuracy of the test dataset or generalization capacity decreases accordingly. It support that function sharing weakens systematic generalization. Please refer to Section 3 for more details.

This paper contributes to uncovering a built-in conflict between deep learning and systematic generalization. We hope this study provides novel insights, forms a basis for new research directions, and helps improve machine intelligence to the human level.

2 A BUILT-IN CONFLICT

We hypothesize a built-in conflict between the function sharing in deep learning and systematic generalization. We cover the definition, assumptions, and derivations of the conflict.

2.1 SYSTEMATIC GENERALIZATION

We have input set X and output set Y. Y contains K factors $Y_1, \ldots, Y_K$, which can be entangled. The training set $\mathcal{D}_{\text{train}}$ has input X_{train} and output Y_{train}. The test set $\mathcal{D}_{\text{test}}$ has input X_{test} and output Y_{test}. In systematic generalization, Y_{train} and Y_{test} are disjoint. However, the values for each factor i are included in the training output. A model f maps input X to the prediction of output $f(X)$. A model enables systematic generalization if it correctly predicts the ground-truth test outputs.

Definition 1 (Systematic generalization)**.** *The dataset requires that any test label is not a training label, but each factor of a test label is seen in a training label.*

$$\forall (x, y) \in \mathcal{D}_{test} : y \notin Y_{train}, \quad \forall i = 1, \ldots, K, \exists y' \in Y_{train} : y'_i = y_i$$

A function f enables systematic generalization if $\forall (x, y) \in \mathcal{D}_{test} : y = f(x)$.

When the model is well trained, we assume it correctly predicts training samples.

Assumption 1 (Correct training prediction)**.** $\forall (x, y) \in \mathcal{D}_{train} : y = f(x)$.

2.2 FUNCTION SHARING

In Figure 2, deep learning prefers the model in (a) over the model in (b). We denote the models as f and g, respectively. We see that g has one more region for the new factor combination between the magenta and cyan boundaries. This region exists because the magenta boundary in g splits the +/+ region in f into two parts. It means the partition under g refines that under f, but not vice versa.

We assume a property of function sharing. For general models f and g, deep learning more or equally prefers f over g if the partition under g is a refinement of that under f. Equivalently, if two inputs have an identical prediction in g, their predictions are still equal in f. It means an equal prediction in g implies that in f.

Assumption 2 (Function sharing)**.** *Deep learning more or equally prefers f over g if*

$$\forall x_a, x_b \in X : g(x_a) = g(x_b) \implies f(x_a) = f(x_b)$$

Note that it is a bias in the learning process. In case we need f to be strictly more preferred over g, we can additionally show that identical prediction in f does not imply that in g.

We consider what causes the function sharing. Intuitively, the preference comes from greedy optimization and the task requirement to learn a complicated model. A model learns to split or partition inputs by different predictions. Some splits (can be a factor) might be easier to learn than others. For example, learning to split inputs by color is more straightforward than by shape. Then the greedy optimization learns it quickly and reuses its intermediate functions to learn other splits. It is not likely that multiple splits (or factors) are learned equally fast during the whole training process.

The mechanism of reusing function is similar to auxiliary tasks. The parameters are updated to address a complicated main task while quickly learning and keeping the prediction ability for more straightforward auxiliary tasks. A more common but less similar example is pre-training. Pre-trained modules, such as image feature extractors or word embeddings, share the internal functions with the main target tasks. However, the pre-training and the main task training do not happen simultaneously. We will discuss more insights in Section 4.

2.3 THE CONFLICT

We derive propositions for proving the theorem and explaining the reasons for the phenomena. In practice, the assumptions may not hold strictly, and the effects come more from the soft biases of the conclusions. The proofs are in Appendix A.

Assumption 2 leads a model to predict training outputs for any input.

Proposition 1 (Seen prediction)**.** *From Assumption 2,* $\forall x \in X : f(x) \in f(X_{train})$.

Informally, suppose there is a non-empty set of all inputs that a function f does not map to training outputs $f(X_{\text{train}})$. In that case, we can design another function f' that predicts a training output for these inputs and keeps predictions for other inputs. Then both f and f' equivalently distinguish training outputs. With Assumption 2, f' is preferred, hence $\forall x \in X : f(x) \in f(X_{\text{train}})$. It does not apply Assumption 1, indicating that the phenomena may happen before a model is well trained.

If a model performs well for the training set, it predicts training ground-truth output for any input.

Proposition 2 (Seen label)**.** *From Assumption 1 and Proposition 1,* $\forall x \in X : f(x) \in Y_{train}$.

It says that the prediction is not any new output. It is a stronger argument than avoiding a particular new output for each input in systematic generalization. It explains that prediction is incorrect because any new output is resisted. We evaluate it in Section 3.

We then look at the conflict. For any test sample (x, y), the definition of systematic generalization requires that the output y is not a training output Y_{train}. However, this contradicts Proposition 2.

Theorem 1 (The conflict)**.** *From Definition 1 and Proposition 2,* $\forall (x, y) \in \mathcal{D}_{test} : y \neq f(x)$.

We covered systematic generalization definition and function sharing assumption. We then derived propositions and a theorem of the conflict.

3 Experiments

We run experiments to show that function sharing reduces the ability of systematic generalization in deep learning. We not only compare sharing or not but also adjust the degree of sharing. We focus on the cases where new outputs are unseen combinations of seen factor values. We cover different standard deep neural network models. The details of networks and experiments can be found in Appendix B. The results of zero-shot learning datasets are in Appendix C, where factors are mainly related to the input locality. We look at the experiment settings and results.

3.1 Settings

Data preparation We construct a dataset from two ten-class classification datasets. The training data are generated from the original training dataset. Y_1 is chosen from all possible labels. Y_2 is chosen from five classes $\{Y_1, Y_1 + 1, \ldots, Y_1 + 4\}$ (we use modular for labels). The test data are generated from the original test dataset. Y_1 is chosen in the same way as in training. Y_2 is chosen from the other classes $\{Y_1 + 5, Y_1 + 6, \ldots, Y_1 + 9\}$. In this design, training and test label combinations are mutually exclusive, but test labels for each output factor are seen in training. Any factor label appears evenly in training and test combinations. X_1 and X_2 are chosen conditioned on their labels Y_1, Y_2, respectively, and merged as the input X. The original datasets and input merge methods vary for each experiment. All the choices follow uniform distributions.

Architecture To evaluate the influence of function sharing on new outputs, we change the function sharing ability while keeping other properties stable for a deep learning model. So we modify function sharing related to new outputs. Since a new output appears only as a new factor combination in our setting, we adjust the sharings between output factors. It does not need to remove all sharings and avoid difficulties in model design. We choose a layer and duplicate the following layers, keeping the number of all hidden nodes in each layer if feasible (Figure 4). We call the former part a shared network and the latter parts individual networks. Each individual network predicts one output factor, so the output is disentangled. We will discuss entangled outputs in Section 4. We keep the depth of the whole network and change the depth of the shared network and individual networks. Note that only sharing the input layer indicates learning two separate models.

Evaluation metrics We use accuracy as the metric. A sample prediction is correct if all the outputs are correct. We have three types of accuracy. The first is the regular evaluation of test data for systematic generalization (a: **Test Sample Accuracy**) corresponding to Theorem 1. We also consider a set of inputs mapped to unseen output combinations corresponding to Proposition 2. We

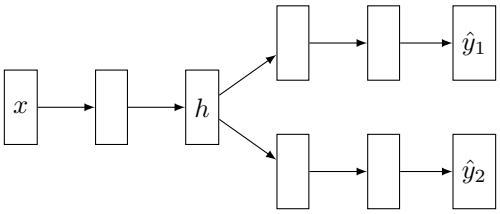


Figure 4: Deep model architecture. x is input, and $f(x) = \hat{y}_1, \hat{y}_2$ are outputs. We duplicate the layers after layer h. The shared network is up to h, and the individual networks are after h (their depths are all two here). We change the shared network depth while maintaining the entire network depth so that the function sharing ability is adjusted while other network properties are stable.

evaluate whether the test samples predict one of the unseen output factor combinations (b: **Test Set Accuracy**). However, test samples are only a subset of input space. If a model learns a different set of factors, the expected inputs may not be those of test samples. So we also evaluate systematic generalization as a model property for any valid input. We randomly draw test inputs from the whole input space (c: **Random Set Accuracy**)[1]. We run each experiment five times and plot the mean and the standard deviation (Figure 5). The result numbers are in Table 1 (Appendix B.3).

$$a: \ \mathbb{E}_{(x,y)\sim P(\mathcal{D}_{\text{test}})}[\delta(f(x) = y)] \quad b: \ \mathbb{E}_{x\sim P(X_{\text{test}})}[\delta(f(x) \in Y_{\text{test}})] \quad c: \ \mathbb{E}_{x\sim U[\mathcal{X}]}[\delta(f(x) \in Y_{\text{test}})]$$

3.2 Results

Fully Connected Network We use an eight-layer fully connected neural network with a flattened image input. We use the Fashion dataset (Xiao et al., 2017) and the MNIST dataset (LeCun et al., 1998). The datasets are uncomplicated to avoid the training data under-fitting for a fully connected neural network. We merge the two inputs by averaging values at each input node.

Convolutional Network We use a convolutional neural network with six convolutional layers and two fully connected layers. We use the CIFAR-10 dataset (Krizhevsky, 2009) and the Fashion dataset (Xiao et al., 2017). We scale the input sizes to that of the larger one and broadcast gray images to colored ones. We merge the inputs by averaging at each node. We use the Fashion dataset as one factor because the average of two colored images can cause training data under-fitting for convolutional neural networks.

Residual Network We use ResNet50 (He et al., 2016), which has five stages, each treated as a layer while changing the shared network depth. It has the dataset setting in the CNN experiment.

Vision Transformer We use Vision Transformer (Dosovitskiy et al., 2021) with one fully connected layer for each patch, five attention layers, and two fully connected layers. We treat the patches as one layer. It has the dataset setting in the CNN experiment.

LSTM A recurrent network has the same parameters for each layer, so it does not support learning different individual networks. Instead, we treat an LSTM as a layer. We use stacked LSTM models with an embedding layer, five bidirectional LSTM layers, and two fully connected layers. We use the Reuters dataset (Joachims, 1998) for both the first and the second datasets. We filter samples by a maximum input length of 200 and use the most frequent ten classes of samples. We merge inputs by concatenating two input texts. The inputs have different lengths because the text lengths vary.

We also run experiments for one-layer LSTM, which compares sharing or not sharing all layers. The results indicate that the shared network has less generalization than the individual network (Table 1).

Transformer We use Transformer (Vaswani et al., 2017). Since it is a classification problem, we only use the encoder. It has one embedding layer, five hidden layers, and two fully connected layers. We use the same dataset setting as the LSTM experiment.

[1] $\delta(\cdot)$ is 1 if the statement is true and 0 otherwise. $U[\mathcal{X}]$ is the uniform distribution of valid inputs.

Summary of results Figure 5 shows that, for each evaluation, the accuracy on the left end (not sharing any hidden layer) is higher than that on the right end (sharing all hidden layers), and it generally decreases as the shared network depth increases. The results indicate that the function sharing weakens systematic generalization.

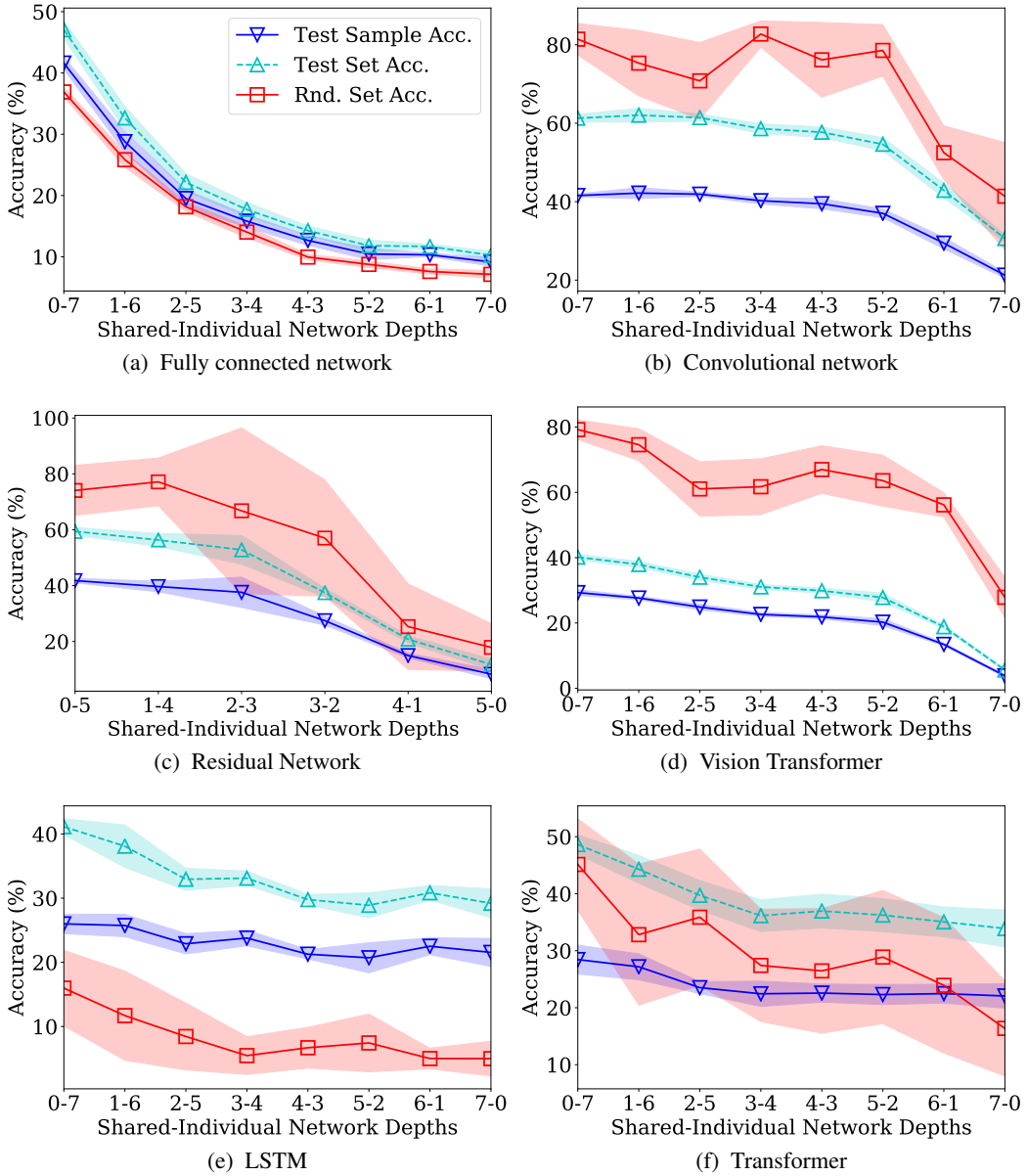


Figure 5: Results of three types of systematic generalization accuracy with shared and individual (sub)network depths. As the shared network depth increases, the accuracy generally decreases, indicating that function sharing reduces systematic generalization ability. Test Set Accuracy and Random Set Accuracy are the ratios of unseen factor combinations in prediction. Test Sample Accuracy and Test Set Accuracy are computed with the test data, and Random Set Accuracy is with randomly generated inputs.

4 DISCUSSIONS

4.1 ENTANGLED OUTPUT

Though the theory does not require disentangled output, the experiments use it to split the network into individual ones. We discuss that entangled output prediction is not easier than disentangled one. Hence the experiment conclusions will likely extend to entangled outputs. (1) Disentangled output is a particular and usually less complicated case of entangled output. (2) Entangled output can be seen as another shared layer, and the experiments show that increasing shared layers reduces systematic generalization ability, so entangled outputs are also likely to suffer from the problem. (3) We can see each output node of an entangled output as a factor, so it becomes disentangled output. The generalization is even more difficult if a value is unseen for an output node.

4.2 BEYOND FACTOR RECOMBINATION

The definition of systematic generalization (Definition 1) requires that each test label factor is seen in a training label. However, it is not directly used in the derivations, and the conclusions may apply to more general o.o.d. problems beyond recombining factors. A new output may correspond to an unseen activation for an output node, e.g., a new class in a classification problem. In such settings, it is sometimes discussed that the bias parameter of the output is a reason to avoid the new value prediction because it does not have any training signal to increase its value. This work provides another reason for not predicting a new value.

4.3 WHY DOES FUNCTION SHARING HAPPEN IN DEEP LEARNING?

We discuss that function sharing can be caused by characteristics of deep learning: deep architecture, shareable network, and greedy optimization. Deep architecture and shareable networks make it possible for factors to share elaborated functions, and greedy optimization encourages the sharing. Deep architecture and greedy search are necessary for deep learning. Deep learning uses deep architecture to fit complicated non-linear functions. Deep learning has a large and complex parameter space. To search in it, we need some prioritization, which leads to a greedy search. The shareable network is widely used in standard deep learning models and works well for i.i.d. problems. However, it is less critical compared to the other ones.

4.4 POTENTIAL SOLUTIONS

We consider possible solutions to avoid function sharing and achieve systematic generalization. From the above discussion, we look at shareable networks. We consider the recombination of factors and focus on sharing between factors. Then, one potential solution uses individual networks for output factors, similar to the experiment setup. We discuss how to design networks when the input or the output is entangled. If the output is entangled, we can design an architecture where each individual network only changes one factor in the output. For example, one individual network changes color, and another changes shape. If the input is entangled, we need to extract factors from it to feed the individual networks. It contains two questions: how to avoid spurious influence from other factors and keep it working in test distribution. We can bottleneck representations for the first one and divide the input into units invariant in training and test for the other, e.g., words or objects.

5 RELATED WORK

Systematic generalization and deep learning Systematic generalization[2] (Fodor & Pylyshyn, 1988; Lake & Baroni, 2018; Bahdanau et al., 2019) is considered the "Great Move" of evolution, caused by the need to process an increasing amount and diversity of environmental information (Newell, 1990). Cognitive scientists see it as central for an organism to view the world (Gallistel & King, 2011). Studies indicate it is related to the prefrontal cortex (Robin & Holyoak, 1995). It was discussed that commonsense is critical (Mccarthy, 1959; Lenat et al., 1986) for systematic generalization, and recent works aim to find general prior knowledge (Goyal & Bengio, 2020), e.g.,

[2]It is also called compositional generalization in other literature.

Consciousness Prior (Bengio, 2017). Levels of systematicity were defined (Hadley, 1992; Niklasson & van Gelder, 1994), and types of tests were summarized (Hupkes et al., 2020). We focus on the primary case with an unseen combination of seen factor values.

A closely related field is causal learning, rooted in the eighteenth-century (Hume, 2003) and classical fields of AI (Pearl, 2003). It was mainly explored from statistical perspectives (Pearl, 2009; Peters et al., 2016; Greenland et al., 1999; Pearl, 2018) with do-calculus (Pearl, 1995; 2009) and interventions (Peters et al., 2016). The causation forms Independent Causal Mechanisms (ICMs) (Peters et al., 2017; Schölkopf et al., 2021). Systematic generalization is the counterfactual when the joint input distribution is intervened to have new values with zero probability in training (covariate shift). This work indicates that standard neural networks do not prefer to learn ICMs.

Parallel Distributed Processing (PDP) models (Rumelhart et al., 1986) use Connectionist models with distributed representations, which describe an object in terms of a set of factors. Though they have the potential to combine the factors to create unseen object representations (Hinton, 1990), it was criticized that they do not address systematic generalization in general (Fodor & Pylyshyn, 1988; Marcus, 1998). Deep learning is a recent PDP model with many achievements (LeCun et al., 2015; He et al., 2016). It was studied that deep neural networks use the composition of functions to achieve high performance (Montufar et al., 2014). The improvements in i.i.d. problems encourage to equip deep learning with systematic generalization.

Recent directions In addition to architecture design (Russin et al., 2019; Andreas et al., 2016) and data augmentation (Andreas, 2020; Akyürek et al., 2021; Jia & Liang, 2016), the main perspectives for systematic generalization approaches include disentangled representation learning, attention mechanism, and meta-learning.

Disentangled representation (Bengio et al., 2013) is learned in unsupervised manners. Early methods learn the representation from statistical independence (Higgins et al., 2017; Locatello et al., 2019). Later, the definition of disentangled representation was proposed with symmetry transformation (Higgins et al., 2018). It leads to Symmetry-based Disentangled Representation Learning (Caselles-Dupré et al., 2019; Painter et al., 2020; Pfau et al., 2020). A disentangled representation learning model can be used as a feature extractor for other systematic generalization tasks.

Attention mechanisms are widely used in neural networks (Bahdanau et al., 2015). Transformers (Vaswani et al., 2017) are modern neural network architectures with self-attention. Recurrent Independent Mechanisms (Goyal et al., 2021b) use attention and the name of the incoming nodes for variable binding. Global workspace (Goyal et al., 2021a) improves them by using limited-capacity global communication to enable the exchangeability of knowledge. Discrete-valued communication bottleneck (Liu et al., 2021) further enhances systematic generalization ability.

Meta-learning (Lake, 2019) usually designs a series of training tasks for learning a meta-learner and uses it in a target task. Each task has training and test data, where test data requires systematic generalization from training data. When ICMs are available, they can be used to generate meta-learning tasks (Schölkopf et al., 2021). Meta-reinforcement learning was used for causal reasoning (Dasgupta et al., 2019). Meta-learning can also capture the adaptation speed to discover causal relations (Bengio et al., 2020; Ke et al., 2019).

Deep learning is a fast-growing field, and many efforts focus on designing architectures and algorithms to improve its performance. However, it is less discussed why standard deep learning models do not achieve systematic generalization. This paper looks into a built-in conflict.

6 Conclusion

This paper investigates a built-in conflict between deep learning and systematic generalization. It explains one of the reasons why standard neural networks seldom achieve systematic generalization. We hypothesize that the conflict is caused by sharing internal functions, and experiments support it. A model partitions an input space into multiple parts separated by boundaries. The function sharing tends to reuse the boundaries, leading to fewer parts for new outputs, which conflicts with systematic generalization. The phenomena are shown in different standard deep neural networks. We hope this finding provides a new understanding of systematic generalization mechanisms in deep learning and helps to improve machine learning algorithms for a higher level of artificial intelligence.

REFERENCES

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PS3IMnScugk.

Jacob Andreas. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7556–7566, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.676. URL https://www.aclweb.org/anthology/2020.acl-main.676.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkezXnA9YX.

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sebastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxWIgBFPS.

Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/36e729ec173b94133d8fa552e4029f8b-Paper.pdf.

Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1785, 2009.

Jerome A. Feldman and Dana H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6(3):205–254, 1982.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Charles R Gallistel and Adam Philip King. *Memory and the computational brain: Why cognitive science will transform neuroscience*. John Wiley & Sons, 2011.

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace, 2021a.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=mLcmdlEUxy-.

Sander Greenland, Judea Pearl, James M Robins, et al. Causal diagrams for epidemiologic research. *Epidemiology*, 10:37–48, 1999.

R. F. Hadley. Compositionality and systematicity in connectionist language learning. In *Conference of the Cognitive Science Society*, 1992.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Irina Higgins, David Amos, David Pfau, Sébastien Racanière, Loïc Matthey, Danilo J. Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *CoRR*, abs/1812.02230, 2018. URL http://arxiv.org/abs/1812.02230.

Geoffrey E. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1):47–75, 1990.

David Hume. *A treatise of human nature*. Courier Corporation, 2003.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? (extended abstract). In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 5065–5069. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/708. URL https://doi.org/10.24963/ijcai.2020/708. Journal track.

Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12–22, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1002. URL https://aclanthology.org/P16-1002.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pp. 137–142, Berlin, Heidelberg, 1998. Springer-Verlag. ISBN 3540644172. doi: 10.1007/BFb0026683. URL https://doi.org/10.1007/BFb0026683.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Bernhard Schölkopf, Michael C Mozer, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pp. 2873–2882, 2018.

Brenden M Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, pp. 9788–9798, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

DB Lenat, M. Prakash, and M. Shepherd. Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *Ai Magazine*, 6(4):65–85, 1986.

Dianbo Liu, Alex M Lamb, Kenji Kawaguchi, Anirudh Goyal ALIAS PARTH GOYAL, Chen Sun, Michael C Mozer, and Yoshua Bengio. Discrete-valued neural communication. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 2109–2121. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/10907813b97e249163587e6246612e21-Paper.pdf.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Gary F Marcus. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

J. Mccarthy. Program with common sense. *Teddington Conference on the Mechanization of Thought Processes*, 1959.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf.

Allen Newell. *Unified Theories of Cognition*. Harvard University Press, USA, 1990. ISBN 0674920996.

Lars Niklasson and Tim van Gelder. Can connectionist models exhibit non-classical structure sensitivity? In *PROCEEDINGS OF THE SIXTEENTH ANNUAL CONFERENCE OF THE COGNITIVE SCIENCE SOCIETY. ERLBAUM*, pp. 664–669, 1994.

Matthew Painter, Adam Prugel-Bennett, and Jonathon Hare. Linear disentangled representations and unsupervised action estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13297–13307. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/9a02387b02ce7de2dac4b925892f68fb-Paper.pdf.

Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2751–2758, 2012.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

Judea Pearl. *CAUSALITY: MODELS, REASONING, AND INFERENCE*. Cambridge university press, 2003.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl. Does obesity shorten life? or is it the soda? on non-manipulable causes. *Journal of Causal Inference*, 6(2), 2018.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

David Pfau, Irina Higgins, Alex Botev, and Sébastien Racanière. Disentangling by subspace diffusion. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17403–17415. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c9f029a6a1b20a8408f372351b321dd8-Paper.pdf.

N. Robin and K. J. Holyoak. Relational complexity and the functions of the prefrontal cortex. *Cognitive Neurosciences*, 1995.

David E. Rumelhart, James L. McClelland, and Au. Parallel distributed processing: Explorations in the microstructure of cognition: Foundations, 1986.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Tristan Sylvain, Linda Petrini, and Devon Hjelm. Locality and compositionality in zero-shot learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hye_V0NKwr.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. In *arXiv:1708.07747*, 2017.

A PROOFS

Proposition 1 (Seen prediction)**.** *From Assumption 2,* $\forall x \in X : f(x) \in f(X_{train})$.

Proof. We are going to prove that for a function g with at least one input mapped to a new output, there exists a more preferred function f with all inputs mapped to seen outputs. Therefore, if a function does not have other functions more preferred over it, the function follows the proposition.

Given a function g, we construct a function f. We pick a $x' \in X_{\text{train}}$. $\forall x \in X$:

$$\begin{aligned} g(x) \in g(X_{\text{train}}): \quad & f(x) = g(x), \\ \text{o.w.}: \quad & f(x) = f(x'). \end{aligned}$$

Then, $\forall x_a, x_b \in X$:

$$\begin{aligned} g(x_a) \in g(X_{\text{train}}): \quad & g(x_a) = g(x_b) \implies f(x_a) = g(x_a) = g(x_b) = f(x_b) \\ \text{o.w.}: \quad & g(x_a) = g(x_b) \implies g(x_b) \notin g(X_{\text{train}}) \implies f(x_a) = f(x') = f(x_b) \end{aligned}$$

In both cases,

$$g(x_a) = g(x_b) \implies f(x_a) = f(x_b)$$

On the other hand,

$$\exists x \in X_{\text{test}} : g(x) \notin g(X_{\text{train}}) \implies g(x) \neq g(x') \in g(X_{\text{train}}), \quad f(x) = f(x').$$

So $f(x) = f(x')$ does not imply $g(x) = g(x')$. With Assumption 2, f is preferred over g.

Also, $\forall x \in X$:

$$\begin{aligned} g(x) \in g(X_{\text{train}}): \quad & \exists x'' \in X_{\text{train}} : g(x) = g(x'') \implies f(x) = f(x'') \in f(X_{\text{train}}) \\ \text{o.w.}: \quad & f(x) = f(x') \in f(X_{\text{train}}) \end{aligned}$$

Therefore, $\forall x \in X : f(x) \in f(X_{\text{train}})$. □

Proposition 2 (Seen label)**.** *From Assumption 1 and Proposition 1,* $\forall x \in X : f(x) \in Y_{train}$.

Proof. From Assumption 1, $f(X_{\text{train}}) = Y_{\text{train}}$. From Proposition 1,

$$\forall x \in X : f(x) \in f(X_{\text{train}}) = Y_{\text{train}}$$

□

Theorem 1 (The conflict)**.** *From Definition 1 and Proposition 2,* $\forall (x, y) \in \mathcal{D}_{test} : y \neq f(x)$.

Proof. $\forall (x, y) \in \mathcal{D}_{\text{test}}$: $y \notin Y_{\text{train}}$ (Definition 1), $x \in X_{\text{test}} \subseteq X \implies f(x) \in Y_{\text{train}}$ (Proposition 2). Therefore $y \neq f(x)$. □

B EXPERIMENT DETAILS

B.1 VISUALIZATION SETTINGS

The model is a fully connected neural network with two input and two output nodes. It has six hidden layers with ReLU activations, and each hidden layer has eight nodes. We use a mini-batch size of 10 with a learning rate of 0.01. We iterate until the model prediction becomes stable. Please see the original work of deep playground for more information. We use six Intel(R) Core(TM) i5-8400 2.80GHz CPUs, and the asset has a public license.

B.2 EXPERIMENT SETTINGS

We use GeForce GTX 1080 or GeForce GTX 1050 Ti GPU for single GPU experiments. We use TensorFlow for implementation. The assets have a public license.

Each input element is linearly scaled to [-0.5, 0.5] for image input. We also uniformly sample from this interval for random image input. We select two sentence lengths uniformly from valid integers (one to maximum length) and then generate each word uniformly from the vocabulary for random text input.

Fully Connected Network The input shape is 28×28, flattened to a vector. There are seven fully connected layers. Each of them has 512 hidden nodes and ReLU activation. The output has ten nodes and Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 512, and we train 2,000 iterations. Each evaluation uses 10,000 samples.

Convolutional Network The input shape is $32 \times 32 \times 3$. There are seven convolutional layers. Each of them has 3×3 kernel size with 64 channels. Then the layer is flattened. We have a fully connected layer with 128 nodes and ReLU activation. The output layer has ten nodes with Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 512, and we train 5,000 iterations. Each evaluation uses 10,000 samples.

Residual Network The input is the same as CNN. The model is the standard ResNet50 implementation. The hidden groups are treated as one layer, so there are five hidden layers. The hidden layer size is 64. The output layer has ten nodes with Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 512, and we train 10,000 iterations. Each evaluation uses 10,000 samples.

Vision Transformer The input is the same as CNN. The model is the standard Vision Transformer implementation with seven hidden layers. The hidden layer size is 64. The output layer has ten nodes with Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 512, and we train 10,000 iterations. Each evaluation uses 10,000 samples.

LSTM The vocabulary size is 30,977, including a start symbol and padding symbol. The input length is 200. The embedding size is 64. There are seven stacked bidirectional LSTM layers, and each has 32 hidden nodes for each direction. Then the output is flattened. The output layer is a fully-connected layer with ten output nodes and Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 64, and we train 1,000 iterations. Each evaluation uses 10,000 samples.

Transformer The input is the same as that of LSTM. The embedding size is 64. There are seven hidden groups. The hidden layer size is 64. The output is flattened. The output layer is a fully-connected layer with ten output nodes and Softmax activation. We use cross-entropy loss and Adam optimizer with a learning rate of 0.001. The batch size is 64, and we train 2,000 iterations. Each evaluation uses 10,000 samples.

B.3 EXPERIMENT RESULTS

We numerically compare the individual (left ends in result figures, 0-max) and shared (right ends, max-0) networks in Table 1. LSTM is the stacked LSTM, and LSTM-1 has only one LSTM layer. It shows that shared network has lower scores than individual network on the three types of accuracy. So it indicates that function sharing avoids systematic generalization.

Table 1: Accuracy (mean $\pm$ std %). Experiment results.

	Test Sample Accuracy		Test Set Accuracy		Random Set Accuracy	
	Individual	Shared	Individual	Shared	Individual	Shared
DNN	41.6 ± 1.0	9.2 ± 0.7	47.1 ± 1.4	10.2 ± 0.6	36.9 ± 0.8	7.1 ± 0.6
CNN	41.6 ± 0.4	21.3 ± 0.8	61.3 ± 1.0	30.6 ± 1.0	81.4 ± 4.0	41.4 ± 13.7
ResNet	41.8 ± 0.8	8.3 ± 1.6	59.4 ± 1.4	11.8 ± 1.9	74.1 ± 9.0	17.9 ± 8.5
ViT	29.3 ± 0.9	3.8 ± 0.4	40.2 ± 0.9	5.6 ± 0.5	79.2 ± 3.0	27.9 ± 6.1
LSTM	26.0 ± 1.5	21.6 ± 2.2	41.1 ± 1.2	29.2 ± 2.3	16.0 ± 5.9	5.0 ± 2.7
LSTM-1	28.6 ± 2.2	26.0 ± 1.9	40.1 ± 2.0	34.8 ± 2.1	15.1 ± 4.1	8.0 ± 2.1
Transformer	28.4 ± 2.5	22.1 ± 2.1	48.6 ± 1.7	33.9 ± 3.2	45.1 ± 8.0	16.4 ± 8.4

C ZERO-SHOT LEARNING DATASETS

We look at the results for Zero-shot learning. We use aPY (Farhadi et al., 2009), AwA2 (Xian et al., 2019), CUB (Wah et al., 2011), and SUN (Patterson & Hays, 2012) datasets. In these datasets, factors are mainly related to the input locality (Sylvain et al., 2020). We use the pre-extracted input features for aPY and AwA and image input for CUB and SUN. We construct output labels from attributes. Each attribute is a binary number, 1 for existing in a sample and 0 otherwise. We select six attributes with the most balanced average number (close to 0.5) in all data. The first, third, and fifth attributes are used for the first output, and the others for the second output. Each output has eight classes yielded from all the combinations of three binary attributes. For aPY, samples may share the same image, so we construct training data from the original training and test data from the original test data. For other datasets, we split all data to disjoint training and test data.

We use an eight-layer CNN model, the same as in the experiment section. The batch size is 512 for aPY and AwA and 256 for CUB and SUN. Other settings are the same as that in the experiment section. The result is shown in Figure 6 and Table 3. Similar to the previous experiments, the depths of shared and individual networks reduce systematic generalization capability.

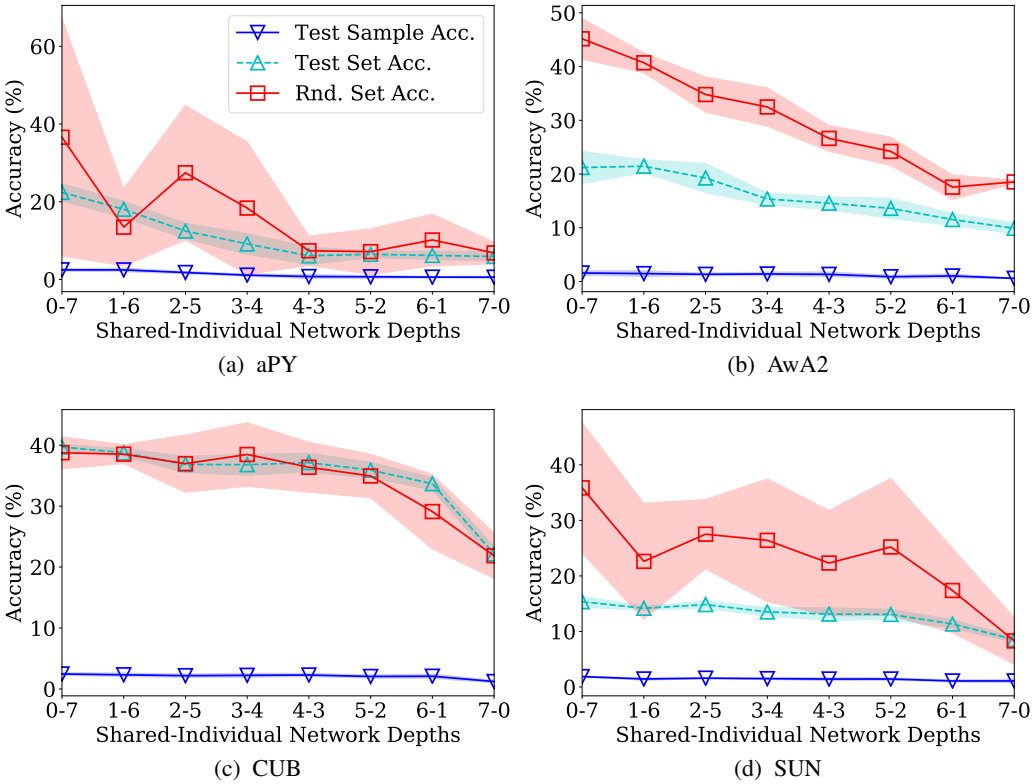


Figure 6: Experiment results for Zero-Shot Learning with CNN models. The results are similar to those in the experiment section.

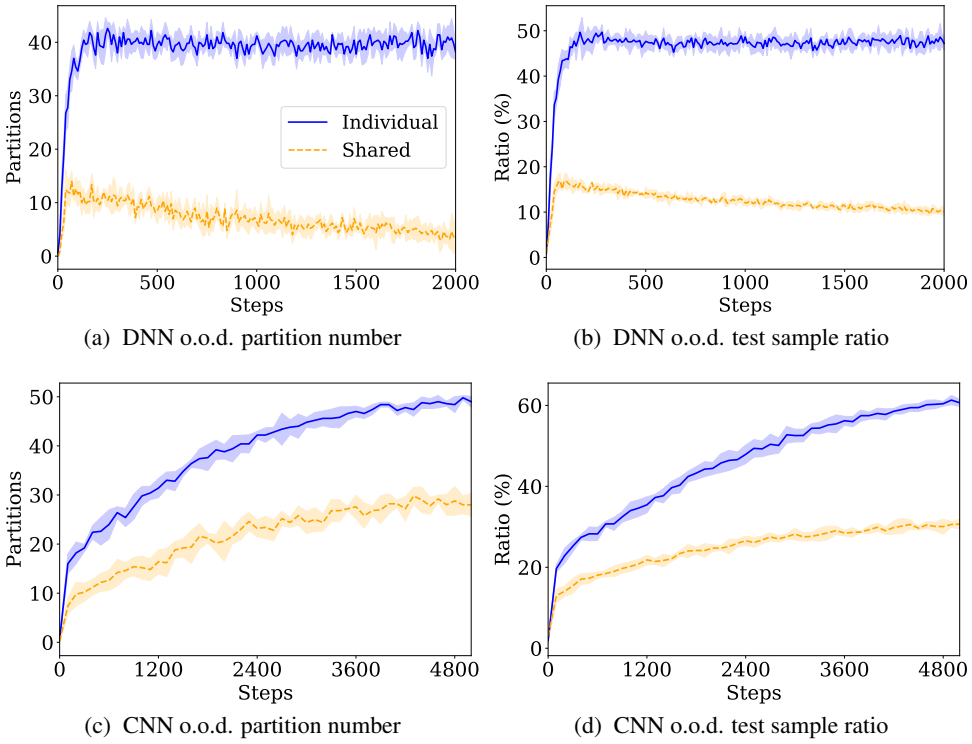
Table 2: Accuracy (mean ± std %). Experiment results for Zero-Shot Learning.

	Test Sample Accuracy		Test Set Accuracy		Random Set Accuracy	
	Individual	Shared	Individual	Shared	Individual	Shared
aPY	2.5 ± 0.3	0.6 ± 0.2	22.4 ± 2.3	5.9 ± 1.2	36.6 ± 30.6	6.8 ± 2.8
AwA2	1.6 ± 0.4	0.6 ± 0.1	21.2 ± 3.0	9.9 ± 1.0	45.2 ± 3.8	18.5 ± 0.3
CUB	2.5 ± 0.2	1.2 ± 0.2	39.7 ± 0.4	22.2 ± 1.2	38.8 ± 2.6	21.9 ± 3.7
SUN	1.9 ± 0.1	1.1 ± 0.3	15.3 ± 1.0	8.5 ± 0.5	35.9 ± 11.7	8.3 ± 4.2

D More Discussions

D.1 Training process

We discussed that sharing boundaries reduces the number of partitions and shrinks the area for new outputs (Proposition 1). We run experiments to find when this happens during training. We sample 10,000 inputs from test data, and if an output combination has at least 50 samples, we regard it as a new output (o.o.d.) partition. We plot the number of o.o.d. partitions and the test sample ratio in the o.o.d. partitions for shared and individual networks in Figure 7. The experiment settings follow DNN and CNN settings in the experiment section. The results show that the differences start to happen in early training.



(a) DNN o.o.d. partition number (b) DNN o.o.d. test sample ratio

(c) CNN o.o.d. partition number (d) CNN o.o.d. test sample ratio

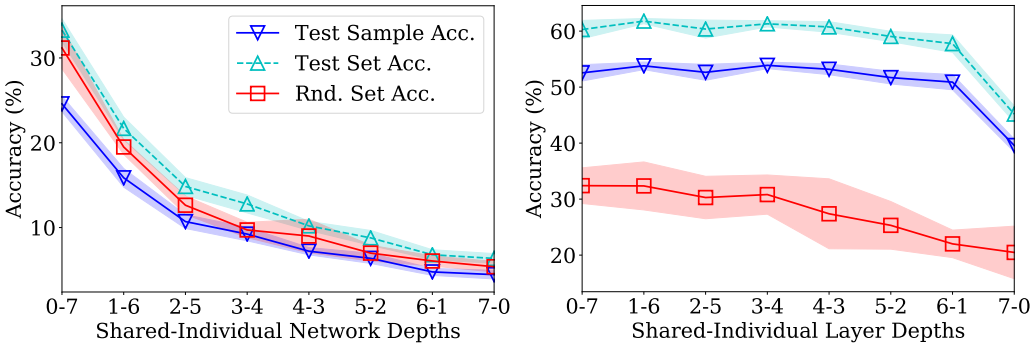
Figure 7: Training process. The o.o.d. partition number and the o.o.d. test sample ratio for shared and individual networks.

D.2 Equally difficult factors

We look at the results when two inputs are equally hard to learn. We use two CIFAR-10 datasets for both the first and the second datasets. Since the average of two colored images can cause training data under-fitting, we merge the inputs by concatenating them by channel. It means the input channel is six. The result is shown in Figure 8 and Table 3. Similar to the previous experiments, when the difficulties are equal, the depth of the shared network weakens systematic generalization.

Table 3: Accuracy (mean $\pm$ std %). Experiment results for equally difficult factors.

	Test Sample Accuracy		Test Set Accuracy		Random Set Accuracy	
	Individual	Shared	Individual	Shared	Individual	Shared
DNN	24.6 ± 0.9	4.4 ± 0.5	33.3 ± 1.3	6.4 ± 0.6	31.2 ± 2.5	5.4 ± 0.6
CNN	52.5 ± 1.5	39.6 ± 1.2	60.3 ± 1.6	45.2 ± 1.5	32.4 ± 3.2	20.5 ± 4.7

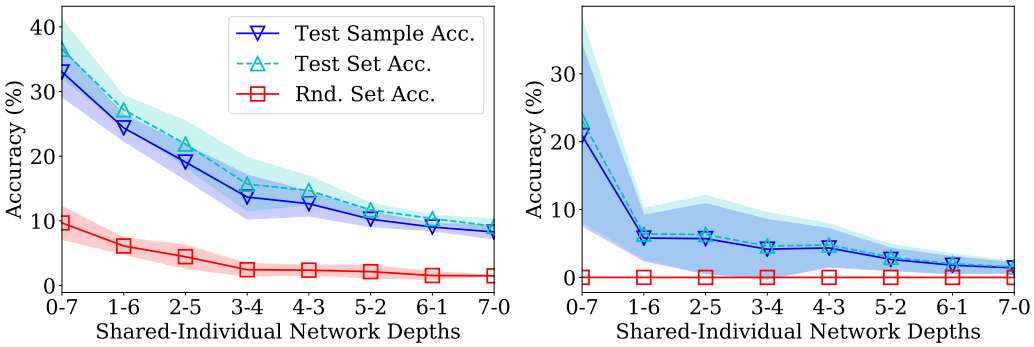


(a) DNN (b) CNN

Figure 8: Experiment results for equally difficult factors. The results are similar to those in the experiment section.

D.3 Label combinations

We also test other training label distribution types. We design tile and one-shot combinations. In tile, a label combination is for training when $Y_1 < 5$ or $Y_2 < 5$. It is similar to the split for illustrative example. In One-shot, a label combination is for training when $Y_1 < 9$ or $Y_2 < 1$. In such a case, only $(9, 0)$ contains $Y_2 = 0$. It is similar to one-shot learning. The results for the fully connected neural network are in Figure 9 and Table 4. It is similar to the results in the experiment section.



(a) Tile (b) One-shot

Figure 9: Experiment results for other label combinations. The results are similar to those in the experiment section.

Table 4: Accuracy (mean ± std %). Experiment results for other label combinations.

	Test Sample Accuracy		Test Set Accuracy		Random Set Accuracy	
	Individual	Shared	Individual	Shared	Individual	Shared
Tile	33.0 ± 3.9	8.3 ± 1.1	36.5 ± 4.6	9.2 ± 1.1	9.7 ± 2.5	1.5 ± 0.1
One-shot	20.9 ± 13.3	1.4 ± 0.8	23.1 ± 14.9	1.5 ± 0.8	0.0 ± 0.0	0.0 ± 0.0