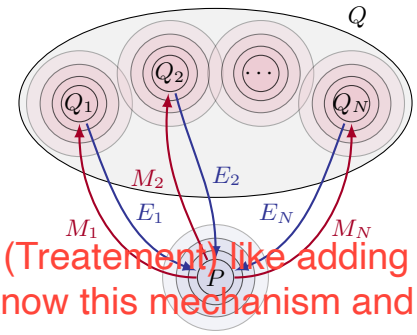
Learning Independent Causal Mechanisms

Giambattista Parascandolo [1 2] **Niki Kilbertus** [1 3] **Mateo Rojas-Carulla** [1 3] **Bernhard Schölkopf** [1]

Abstract

Statistical learning relies upon data sampled from a distribution, and we usually do not care what actually generated it in the first place. From the point of view of causal modeling, the structure of each distribution is induced by physical mechanisms that give rise to dependences between observables. Mechanisms, however, can be meaningful autonomous modules of generative models that make sense beyond a particular entailed data distribution, lending themselves to transfer between problems. We develop an algorithm to recover a set of independent (inverse) mechanisms from a set of transformed data points. The approach is unsupervised and based on a set of experts that compete for data generated by the mechanisms, driving specialization. We analyze the proposed method in a series of experiments on image data. Each expert learns to map a subset of the transformed data back to a reference distribution. The learned mechanisms generalize to novel domains. We discuss implications for transfer learning and links to recent trends in generative modeling.



M1 is causal in terms of generated by specific (Treatement) like adding noise.
They are absolutely causal because we already know this mechanism and we did it.

Figure 1. An overview of the problem setup. Given a sample from a canonical distribution P, and one from a mixture of transformed distributions Q_i obtained by mechanisms M_i on P, we want to learn inverse mechanisms E_i as independent modules. Modules (or *experts*) compete amongst each other for data points, encouraging specialization.

1. Introduction

Humans are able to recognize objects such as handwritten digits based on distorted inputs. They can correctly label translated, corrupted, or inverted digits, without having to relearn them from scratch. The same applies for new objects, essentially after having seen them once. Arguably, human intelligence utilizes *mechanisms* (such as translation) that are independent from an input domain and thus generalize across object classes. These mechanisms are *modular, reusable and broadly applicable*, and the problem of learning them from data is fundamental for the study of transfer and domain adaptation.

In the field of causality, the concept of independent mechanisms plays a central role both on the conceptual level and, more recently, in applications to inference. The *independent mechanisms (IM)* assumption states that the causal generative process of a system's variables is composed of autonomous modules that do not inform or influence each other (Schölkopf et al., 2012; Peters et al., 2017).

If a joint density is Markovian with respect to a directed graph $\mathcal{G}$, we can write it as

$$p(\mathbf{x}) = p(x_1, \dots, x_d) = \prod_{j=1}^{d} p(x_j | \mathrm{pa}_{\mathcal{G}}^j), \qquad (1)$$

where $\mathrm{pa}_{\mathcal{G}}^j$ denotes the parents of variable x_j in the graph.

For a given joint density, there are usually many decompositions of the form (1), with respect to different graphs. If $\mathcal{G}$ is a *causal* graph, i.e., if its edges denote direct causation (Pearl, 2000), then the conditional $p(x_j|\mathrm{pa}_{\mathcal{G}}{}^j)$ can be thought of as physical *mechanism* generating x_j from its parents, and we refer to it as a *causal conditional*. In this case, we consider the factorization (1) a *generative* model where the term "generative" truly refers to a physical generative process. As an aside, we note that in the alter-

[1]Max Planck Institute for Intelligent Systems [2]Max Planck ETH Center for Learning Systems [3]University of Cambridge. Correspondence to: Giambattista Parascandolo <gparascandolo@tue.mpg.de>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

native view of causal models as structural equation models, each of the causal conditionals corresponds to a functional mapping and a noise variable (Pearl, 2000).

By the IM assumption, the causal conditionals are autonomous modules that do not influence or inform each other. This has multiple consequences. First, knowledge of one mechanism does not contain information about another one (Appendix D). Second, if one mechanism changes (e.g., due to distribution shift), there is no reason that other mechanisms should also change, i.e., they tend to remain *invariant*. As a special case, it is (in principle) possible to locally *intervene* on one mechanism (for instance, by setting it to a constant) without affecting any of the other modules. In all these cases, most of (1) will remain unchanged. However, since the overall density will change, most generic (non-causal) conditionals would change.

The IM assumption can be exploited when performing causal structure inference (Peters et al., 2017). However, it also has implications for machine learning more broadly. A model which is expressed in terms of causal conditionals (rather than conditionals with respect to some other factorization) is likely to have components that better transfer or generalize to other settings (Schölkopf et al., 2012), and its modules are better suited for building complex models from simpler ones. Independent mechanisms as sub-components can be trained independently, from multiple domains, and are more likely to be re-usable. They may also be easier to *interpret* and provide more *insight* since they correspond to physical mechanisms. this is key-idea

Animate intelligence cannot afford to learn new models from scratch for every new task. Rather, it is likely to rely on robust local components that can flexibly be re-used and re-purposed. It also requires local mechanisms for adapting and training modules rather than re-training the whole brain every time a new task is learned. Currently, machine learning excels at optimizing well-defined tasks from large i.i.d. datasets. However, if we want to move towards life-long learning and generalization across tasks, then we need to understand how modules can be learnt from data and shared between tasks.

In the present paper, we focus on a class of such modules, and on algorithms to learn them from data. We describe an architecture using competing experts that automatically specialize on different image transformations. The resulting model is attractive for lifelong learning, with the possibility of easily adding, removing, retraining, and re-purposing its components independently. It is unsupervised in the sense that the images are not labeled by the transformations they have undergone. We only need a sample from a reference distribution and a set of transformed images. The transformed images are based on another sample, and no pairing or information about the transformations is available.

We test our approach on *MNIST* digits which have undergone various transformations such as contrast inversion, noise addition and translation. Information about the nature and number of such transformations is not known at the beginning of training. We identify the independent mechanisms linking the reference distribution to a distribution of modified digits, and learn to invert them without supervision.

The inverse mechanisms can be re-purposed as preprocessors, to transform modified digits which are subsequently classified using a standard MNIST classifier. The trained experts also generalize to *Omniglot* characters, none of which were seen during training. These are promising results pointing towards a form of robustness that animate intelligence excels at.

*still, it's not solving underspecified problem

2. Related work

Our work mainly draws from mixtures of experts, domain adaptation, and causality.

Early works on mixture of experts date back to the early nineties (Jacobs et al., 1991; Jordan & Jacobs, 1994), and since then the topic has been subject of extensive research. Recent work includes that of Shazeer et al. (2017), successfully training a mixture of 1000 experts using a gating mechanism that selects only a fraction of experts for each example. Aljundi et al. (2017) train a network of experts on multiple tasks, with a focus on lifelong learning; autoencoders are trained for each task and used as gating mechanisms. Lee et al. (2016) propose Stochastic Multiple Choice Learning, an algorithm which resembles the one we describe in Section 3, aimed at training mixture of experts to propose a diverse set of outputs. The main differences are that our model is trained jointly with a *learned* selection system which is valid also at test time, that our trained experts learn independent mechanisms and can be combined (cf. Figure 8), and in the way experts are initialized.

Another research direction that is relevant to our work is unsupervised domain adaptation (Bousmalis et al., 2017). These methods often use some supervision from labeled data and/or match the two distributions in a learned feature space (e.g. Tzeng et al., 2017).

The novelty of our work lies in the following aspects: (1) we automatically identify and invert a set of independent (inverse) causal mechanisms; (2) we do so using only data from an original distribution and from the mixture of transformed data, without labels; (3) the architecture is modular, can be easily expanded, and its trained modules can be reused; and (4) the method relies on competition of experts.

Ideas from the field of causal inference inspire the present work. Understanding the data generating mechanisms plays

a key role in causal inference, and goes beyond the statistical assumptions usually exploited in machine learning. Causality provides a framework for understanding how a system responds to *interventions*, and causal graphical models as well as structural equation models (SEM) are common ways of describing causal systems (Pearl, 2000; Peters et al., 2017). The IM assumption discussed in the introduction can be used for identification of causal models (Daniušis et al., 2010; Zhang et al., 2015), but causality has also proven a useful tool for discussing and understanding machine learning in the non-i.i.d. regime. Recent applications include semi-supervised learning (Schölkopf et al., 2012) and transfer learning (Rojas-Carulla et al., 2015), in which the authors focus only on linear regression models. We seek to extend applications of causal inference to more complex settings and aim to learn causal mechanisms and ultimately causal SEMs without supervision.

On the conceptual level, our setting is related to recent work on deep learning for disentangling factors of variation (Chen et al., 2016; Higgins et al., 2017) as well as non-linear ICA (Hyvarinen & Morioka, 2016). In our work, causal mechanisms play the role of factors of variation. The main difference is that we recover mechanisms as independent modules.

3. Learning causal mechanisms as independent modules

The aim of this section is twofold. First, we describe the generative process of our data. We start with a distribution P that we will call "canonical" and an a priori *unknown* number of independent mechanisms which act on (examples drawn from) P. At training time, a sample from the canonical distribution is available, as well as a dataset obtained by applying the mechanisms to (unseen) examples drawn from P. Second, we propose an algorithm which recovers and learns to invert the mechanisms in an unsupervised fashion.

3.1. Formal setting

Consider a canonical distribution P on $\mathbb{R}^d$, e.g., the empirical distribution defined by MNIST digits on pixel space. We further consider N measurable functions $M_1, \dots, M_N : \mathbb{R}^d \to \mathbb{R}^d$, called *mechanisms*. We think of these as independent causal mechanisms in nature, and their number is a priori unknown. A more formal definition of independence between mechanisms is relegated to Appendix D. The mechanisms give rise to N distributions $Q_1, \dots, Q_N$ where $Q_j = M_j(P)$.[1] This setup is illustrated in Figure 1. In the MNIST example, we consider translations or adding noise as mechanisms, i.e., the corresponding Q distributions are translated and noisy MNIST digits.

At training time, we receive a dataset $\mathcal{D}_Q = (x_i)_{i=1}^n$ drawn i.i.d. from a mixture of $Q_1, \dots, Q_N$, and a dataset $\mathcal{D}_P$ sampled independently from the canonical distribution P. Our goal is to identify the underlying mechanisms $M_1, \dots, M_N$ and learn approximate inverse mappings which allow us to map the examples from $\mathcal{D}_Q$ back to their counterpart in P.

If we were given distinct datasets $\mathcal{D}_{Q_j}$ each drawn from Q_j, we could individually learn each mechanism, resulting in independent (approximations of the) mechanisms regardless of the properties of the training procedure. This is due to the fact that the datasets are drawn from independent mechanisms in the first place, and the separate training procedure cannot generate a dependence between them. This statement does not require that the procedure is successful, i.e., that the obtained mechanisms approximate the true M_j in some metric.

In contrast, we do not require access to the distinct datasets. Instead we construct a larger set $\mathcal{D}_Q$ by first taking the union of the sets D_{Q_j}, and then applying a random permutation. This corresponds to a dataset where each element has been generated by one of the (independent) mechanisms, but we do not know by which one. Clearly, it should be harder to identify and learn independent mechanisms from such a dataset. We next describe an approach to handle this setting.

3.2. Competitive learning of independent mechanisms

The training machine is composed of N' parametric functions $E_1, \dots, E_{N'}$ with distinct trainable parameters $\theta_1, \dots, \theta_{N'}$. We refer to these functions as the *experts*. Note that we do not require $N' = N$, since the real number of mechanisms is unknown a priori. The goal is to maximize an objective function $c : \mathbb{R}^d \to \mathbb{R}$ with the key property that c takes high values on the support of the canonical distribution P, and low values outside. Note that c could be a parametric function, and its parameters could be jointly optimized with the experts during training. Below, we specify the details of this rather general definition.

During training, the experts compete for the data points. Each example x' from $\mathcal{D}_Q$ is fed to all experts independently and in parallel. Comparing the outputs of all experts $c_j = c(E_j(x'))$, we select the winning expert E_{j^*}, where $j^* = \arg\max_j(c_j)$. Its parameters θ_{j^*} are updated such as to maximize $c(E_{j^*}(x'))$, while the other experts remain unchanged. The motivation behind competitively updating only the winning expert is to enforce specialization; the best performing expert becomes even better at mapping x' back to the corresponding example from the canonical distribution. We will describe below that alongside with the expert's parameters, we train parameters of

[1] Each distribution Q_j is defined as the pushforward measure of P induced by M_j.

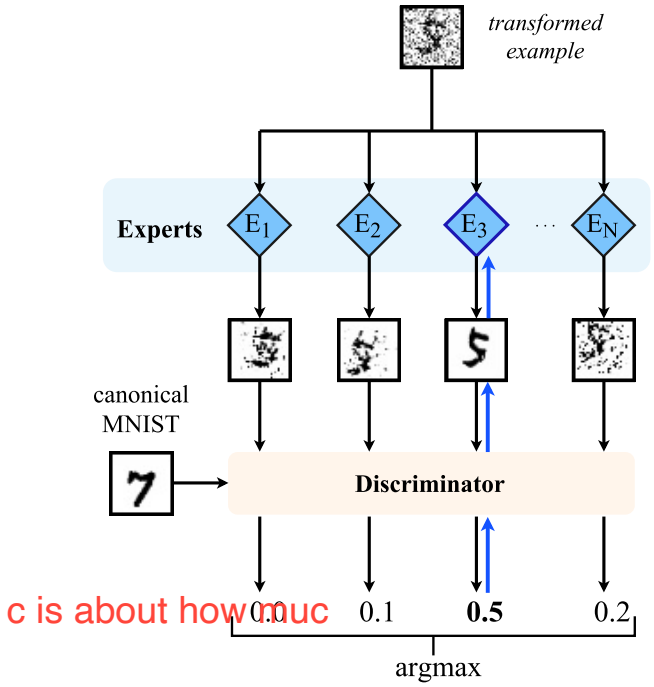


Figure 2. We show how a transformed example, here a noisy digit, is processed by a competition of experts. Only Expert 3 is specializing on denoising, it wins the example and gets trained on it, whereas the others perform translations and are not updated.

c (which in our experiments will be carried in an adversarial fashion). Figure 2 depicts this procedure. Overall, our optimization problem reads:

$$\theta_1^*,...,\theta_{N'}^* = \underset{\theta_1,...,\theta_{N'}}{\text{argmax}} \mathbb{E}_{x' \sim Q} \left(\max_{j \in \{1,...,N'\}} c(E_{\theta_j}(x')) \right). \quad (2)$$

The training described above raises a number of questions, which we address next.

1. Selecting the appropriate number of experts. Generally, the number of mechanisms N which generated the dataset $\mathcal{D}_Q$ is not available a priori. Therefore, we require an adaptive procedure to choose the number of experts N'. This is one of the challenges shared with most clustering techniques. Given the modular behavior of the procedure, experts may be added or removed during or after training, making the framework very flexible. Assuming however that the number of experts is fixed, the following behaviors could occur.

If $N' > N$ (too many experts): a) some of the experts do not specialize and do not win any example in the dataset; or b) some tasks are divided between experts (for instance, each expert can specialize in a mode of the distribution of the same task). In a), the inactive experts can be removed, and in b) experts sharing the same task can be merged into a wider expert.[2]

If $N' < N$ (too few experts): a) some of the experts specialize in multiple tasks or b) some of the tasks are not learned by the experts, so that data points from such tasks lead to a poor score across all experts. We provide experiments substantiating these claims in appendix A.1.

2. Convergence criterion. Since the problem is unsupervised, there is no straightforward way of measuring convergence, which raises the question of how to choose a stopping time for the competitive procedure. As an example, one may act according to one of the following: *a*) fix a maximum number of iterations or *b*) stop if each example is assigned to the same experts for a pre-defined number of iterations (i.e., each expert consistently wins the same data points).

3. Time and space complexity. Each example has to be evaluated by all experts in order to assign it to the winning expert.While this results in a computational cost that depends linearly on the number of experts, these evaluations can be done in parallel and therefore the time complexity of a single iteration can be bounded by the complexity to compute the output of a single expert. Moreover, as each expert will in principle have a smaller architecture than a single large network, the committee of experts will typically be faster to execute.

Concrete protocol for neural networks. One possible model class for the experts are deep neural networks. Training using backpropagation is particularly well suited for the online nature of the training proposed: after an expert wins a data point x', its parameters are updated by backpropagation, while the other experts remain untouched. Moreover, recent advances in generative modeling give rise to natural choices for the loss function c. For instance, through adversarial training (Goodfellow et al., 2014), one can use as objective function the output of a discriminator network trained on the canonical sample $\mathcal{D}_P$ and against the outputs of the experts. In the next section we introduce a formal description of a training procedure based on adversarial training in Algorithm 1, and empirically evaluate its performance.

While in this work we focus on adversarial training, preliminary experiments have shown that similar results can be achieved for example with variational autoencoders (VAE) (Kingma & Welling, 2013). Given a VAE trained on the canonical distribution P, one may define $c(x')$ as the opposite of the VAE loss.

[2]However, note that in order to do this, it is necessary to first acknowledge that the two experts have learned part of the same task, which would require extra information or visual inspection.

4. Experiments

In this set of experiments we test the method presented in Section 3 on the MNIST dataset transformed with the set of mechanisms described in detail in the Appendix C, i.e., eight directions of translations by 4 pixels (up, down, left, right, and the four diagonals), contrast inversion, addition of noise, for a total of 10 transformations. We split the training partition of MNIST in half, and transform all and only the examples in the first half; this ensures that there is no matching ground truth in the dataset, and that learning is unsupervised. As a preprocessing step, the digits are zero-padded so that they have size 32×32 pixels, and the pixel intensities are scaled between 0 and 1. This is done even before any mechanism is applied. We use neural networks for both the experts and the selection mechanism, and employ an adversarial training scheme.

Each expert E_i can be seen as a generator from a GAN conditioned on an input image rather than (as usually) a noise vector. A discriminator D provides gradients for training the experts and acts also as a selection mechanism c: only the expert whose output obtains the higher score from D *wins* the example, and is trained on it to maximize the output of D. We describe the exact algorithm used to train the networks in these experiments in Algorithm 1. The discriminator is trained to maximize the following cross-entropy loss:

$$\max_{\theta_D} \Bigg(\mathbb{E}_{x\sim P} \log(D_{\theta_D}(x)) + \frac{1}{N'} \sum_{j=1}^{N'} \mathbb{E}_{x'\sim Q} \left(\log(1 - D_{\theta_D}(E_{\theta_j}(x'))) \right) \Bigg) \tag{3}$$

For simplicity, we assume for the rest of this section that the number of experts N' equals the number of true mechanisms N. Results where $N \neq N'$ are relegated to Appendix A.1.

Neural nets details. Each expert is a CNN with five convolutional layers, 32 filters per layer of size 3×3, ELU (Clevert et al., 2015) as activation function, batch normalization (Ioffe & Szegedy, 2015), and zero padding. The discriminator is also a CNN, with average pooling every two convolutional layers, growing number of filters, and a fully connected layer with 1024 neurons as last hidden layer. Both networks are trained using Adam as optimizer (Kingma & Ba, 2014), with the default hyper-parameters.[3]

Unless specified otherwise, after a random weight initialization we first train the experts to approximate the identity mapping on our data, by pretraining them on predicting identical input-output pairs randomly selected from the transformed dataset. This makes the experts start from similar grounds, and we found that this improved the speed and robustness of convergence. We will refer to this as *approximate identity initialization* for the rest of the paper.

[3]For the exact experimental parameters and architectures see the Appendix B or the PyTorch implementation we will release.

Algorithm 1 Learning independent mechanisms using competition of experts and adversarial training

Precondition: X: data sampled from P; X': data sampled from $\mathcal{D}_Q$; D discriminator; N': number of experts; T: maximum number of iterations;
(p) highlights that the steps in the instruction can be executed in parallel

▷ Initialize experts as approximately identity **(p)**:
$\{E_i \leftarrow \text{TrainAsIdentityOn}(X')\}_{j=1}^{N'}$
for $t \leftarrow 1$ to T **do**
  ▷ Sample minibatches:
  $x, x' \leftarrow \text{Sample}(X), \text{Sample}(X')$
  ▷ Scores from D for all outputs from the experts **(p)**:
  $\{c_j \leftarrow D(E_j(x'))\}_{j=1}^{N'}$
  ▷ Update D **(p)**:
  $\theta_D^{t+1} \leftarrow \text{Adam}\Big(\theta_D^t, \nabla \log D(x) + \nabla(1/N' \sum_{j=1}^{N'} \log(1 - c_j))\Big)$
  ▷ Update experts **(p)**:
  $\{\theta_{E_j}^{t+1} \leftarrow \text{Adam}(\theta_{E_j}^t, \nabla \max_{j\in 1,\ldots,N'} \log(c_j))\}_{j=1}^{N'}$

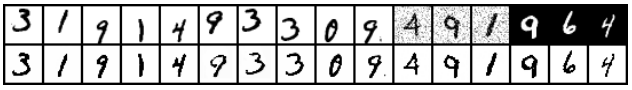


Figure 3. The top row contains 16 random inputs to the networks, and the bottom row the corresponding outputs from the highest scoring experts against the discriminator after 1000 iterations.

A minibatch of 32 transformed MNIST digits, each transformed by a randomly chosen mechanism, is fed to all experts E_i. The outputs are fed to the discriminator D, which computes a score for each of them. For each example the cross entropy loss in Equation (3) and the resulting gradients are computed only for the output of the highest scoring expert, and they are used to update both the discriminator (when 0 is the target in the cross entropy) and the winning expert (when using 1 as the target). In order to further support the winning expert, we punish the losing experts by training the discriminator against their outputs as well. Then, a minibatch of canonical MNIST digit is used in order to update the discriminator with 'real' data. We refer to the above procedure as one *iteration*.

We ran the experiments 10 times with different random seeds for the initializations. Each experiment is run for 2000 iterations.

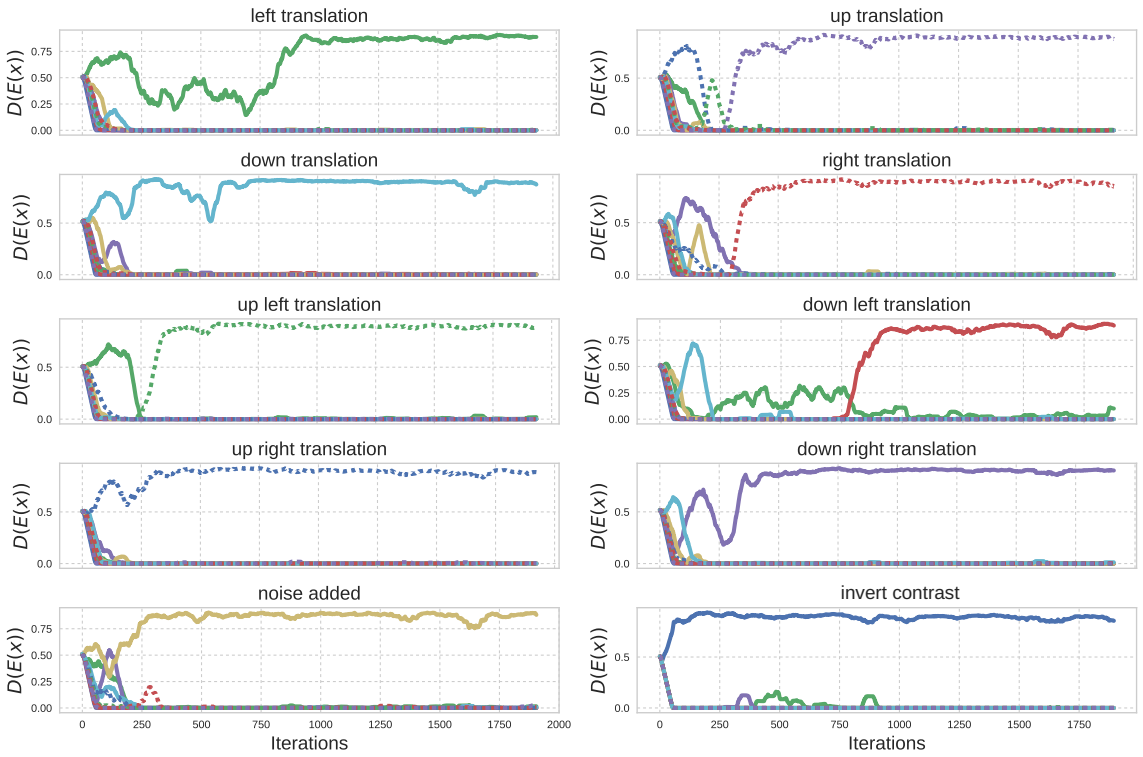


Figure 4. Experts' performance, measured by discriminator scores. Each line color/style represents one expert. For each of ten different mechanisms (top left to bottom right), the experts are being fed transformed digits. Each expert learns to specialize on a different mechanism, as shown by the score approaching 1. Each curve is smoothed with a moving average of 50 iterations.

5. Results

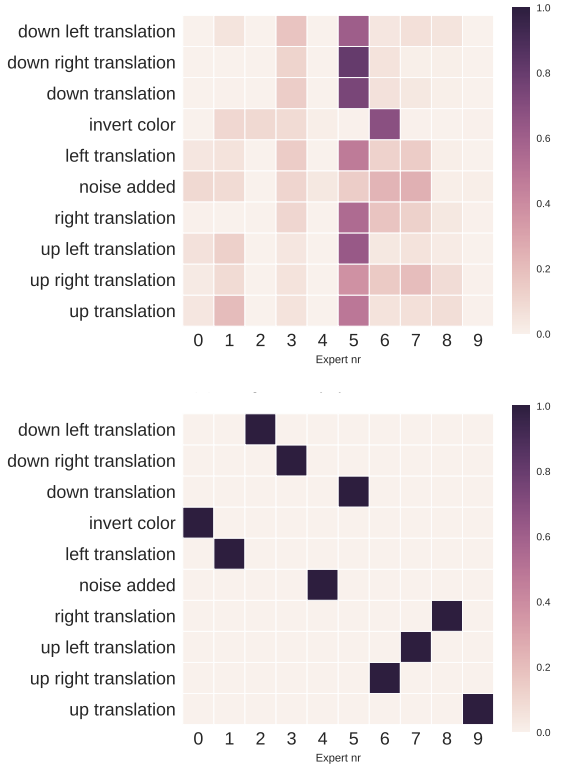
The experts correctly specialized on inverting exactly one mechanism each in 7 out of the 10 runs; in the remaining 3 runs the results were only slightly suboptimal: one expert specialized on two tasks, one expert did not specialize on any, and the remaining experts still specialized on one task each, thus still covering all the existing tasks. In Figure 3 we show a randomly selected batch of inputs and corresponding outputs from the model. Each independent mechanism was inverted by a different expert.

We first discuss our three main findings, and then move on to additional experiments.

1. The experts specialize w.r.t. c. In Figure 4, we plot the scores assigned by the discriminator for each expert on each task in a typical successful run. Each expert is represented with the same color and linestyle across all tasks. The figure shows that after an initial phase of heavy competition, the experts exhibit the desired behavior and obtain a high score on D on one mechanism each. Note how the green expert tries to learn two similar tasks until iteration 750 (left and left-down translation), at which point the red expert takes over one of the tasks. Subsequently, both specialize rapidly. Figure 5 provides further evidence, by visualizing that the assignments of data points to experts induced by c are meaningful. We report the proportion of examples from each task assigned to each expert at the beginning and at the end of training: at first, the assignment of experts to tasks by the discriminator is almost uniform; by the end of the training, each expert wins almost all examples coming from one transformation, and no others.

2. The transformed outputs improve a classifier. In order to test if the committee of experts can recover a good approximation of the original digits, we test the output of our experts against a pretrained standard MNIST classifier. For this, we use the test partition of the data. We compare the accuracy for three inputs: a) the test digits transformed by the mechanisms, b) the transformed digits after being processed by the highest scoring experts (which tries to invert the mechanisms), c) the original test digits. The latter can be seen as an upper bound to the accuracy that can be achieved.

As shown by the two dashed horizontal lines in Figure 6, the transformed test digits achieve a 40% accuracy when tested directly on the classifier, while the untransformed digits would achieve $\approx 99\%$ accuracy. The accuracy for the output digits starts at 40% — due to the identity initialization



(b) After 1000 iterations

Figure 5. The proportion of data won by each expert for each transformation on the digits from the test set.

of the experts — but it subsequently quickly approaches the performance on the original digits as it is trained. Note that after about 600 iterations, i.e., once the networks have seen about one third of the whole dataset *once*, the accuracy has almost reached the upper bound.

3. The experts learn mechanisms that generalize. Given that towards the end of training, each expert E_i is updated only on data points from Q_i, one could imagine that they will not perform well on data points from other distributions. In fact this is not the case. Not only do all experts E_i generalize to all other transformed distributions Q_j, but also to different datasets all together. To show this, we use the Omniglot dataset of letters from different alphabets (Lake et al., 2015) and rescale them to 46×46 pixels (instead of 32×32 of MNIST, which is not an issue since the experts are fully convolutional). We transform a random sample with all mechanisms M_i and test each on all experts E_i, which have only been trained on MNIST. As shown in Figure 7, each network consistently applies the same transformation also on inputs outside of the domain they have specialized on. They indeed learn a *mechanism* that is independent of the input.

Having made our main points, we continue with a few more observations.

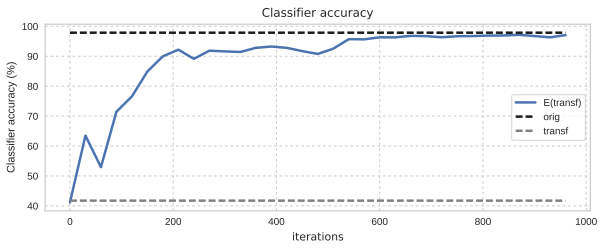


Figure 6. Accuracy of a pretrained CNN MNIST classifier on transformed test digits $\mathcal{D}_Q$, on the same digits after going through our model, and on the original digits. Our system manages to invert the transformations, with the classifier accuracy quickly approaching the optimum. Note that 600 iterations correspond to having seen about a third of the dataset.

The learned inverse mechanisms can be combined. We test whether the trained experts could in principle be used to undo several transformations applied at once, even though the training set consisted only of images transformed by a single mechanism. For simplicity, we assume we know which transformations were used. In Figure 8, we test on Omniglot letters transformed with three consecutive transformations (noise, up left translation, contrast inversion) by applying the corresponding experts previously trained on MNIST, and correctly recover the original letters.

Effect of the approximate identity initialization. For the same experiments but without the approximate identity initialization, several experts fail to specialize. Out of 10 new runs with random initialization, only one experiment had arguably good results, with eight experts specializing on one task each, one on two tasks, and the last one on none. The performance was worse in the remaining runs. The problem was not that the algorithm takes longer to converge following a random initialization, as with an additional experiment for 10 000 iterations the results did not improve. Instead, the random initialization can lead to one expert winning examples from many tasks at the beginning of training, in which case it is hard for the others to catch up.

A simple single-net baseline. Training a single network instead of a committee of experts makes the problem more difficult to solve. Using identical training settings, we trained a single network once with 32, once with 64, and once with 128 filters per layer, and none of them managed to correctly learn more than one inverse mechanism.[4] Note that a single network with 128 filters per layer has about twice as many parameters overall as the committee of 10 experts with 32 filters per layer each. We also tried *a*) random initialization instead of the approximate identity, *b*) reducing

[4] Specifically, the network performs well on the contrast inversion task, and poorly on all others.

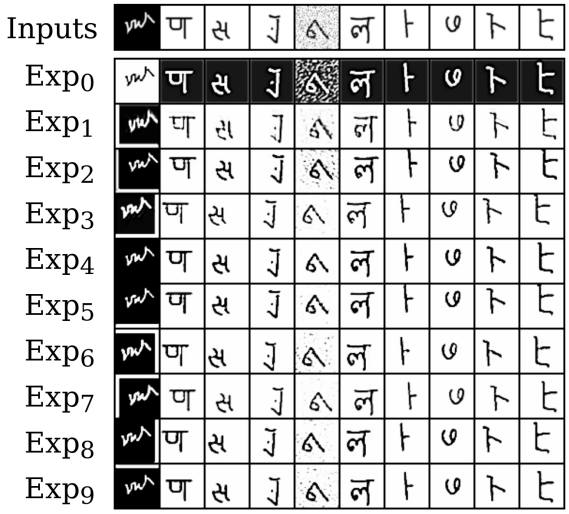


Figure 7. Each column shows how each expert transforms the input presented on top. We arrange the tasks such that the diagonal contains the highest scoring expert for the input given at the top of the column. The experts have learned the inverse mechanisms, consistently applying them to previously unseen symbols.

the learning rate of the discriminator by a factor of 10, and *c*) increasing the receptive field by adding two pooling and two upsampling layers, without any improvement. While we do not exclude that careful hyperparameter tuning may enable a single net to learn multiple mechanisms, it certainly was not straightforward in our experiments.

Specialization occurs also with higher capacity experts. While in principle with infinite capacity and data, a single expert could solve all tasks simultaneously, in practice limited resources and the proposed training procedure favor specialization in independent modules. Increasing the size of the experts from 32 filters per layer to 64 or 128 filters,[5] or enlarging the overall receptive field by using two pooling and two upsampling layers, still resulted in good specialization of the experts, with no more than two experts specializing on two tasks at once.

Fewer examples from the canonical distribution. In some applications, we might only have a small sample from the original distribution. Interestingly, if we reduce the number of examples from the original distribution from 30 000 down to 64, we find that all experts still specialize and recover good approximations of the inverse mechanisms, using the exact same training protocol. Although the output digits turn out less clean and sharp, we still achieve 96% accuracy on the pretrained MNIST classifier.

[5]Equivalent to an increase of parameters from ∼27K to ∼110K or ∼440K parameters respectively.

Figure 8. First row: input Omniglot letters that were transformed with noise, contrast inversion and translation up left. Second to fourth row: application of denoising, contrast inverting and right down translating experts. Last row: ground truth. Although the experts were not trained on a combination of mechanisms nor on Omniglot letters, they can be used to recover the original digits.

6. Conclusions

We have developed a method to identify and learn a set of independent causal mechanisms. Here these are *inverse* mechanisms, but an extension to forward mechanisms appears feasible and worthwhile. We reported promising results in experiments using image transformations; future work could study more complex settings and diverse domains. The method does not explicitly minimize a measure of dependence of mechanisms, but works if the data generating process contains independent mechanisms in the first place: As the different tasks (mechanisms) do not contain information about each other, improving on one of them does not improve performance on another, which is exactly what encourages specialization.

A natural extension of our work is to consider independent mechanisms that *simultaneously* affect the data (e.g., lighting and position in a portrait), and to allow multiple passes through our committee of experts to identify local mechanisms (akin to Lie derivatives) from more complex datasets — for instance, using recurrent neural networks that allow the application of multiple mechanisms by iteration. With many experts, the computational cost (or parallel processing) might become unnecessarily high. This could be mitigated by hybrid approaches incorporating gated mixture of experts or a hierarchical selection of competing experts.

We believe our work constitutes a promising connection between causal modeling and deep learning. As discussed in the introduction, causality has a lot to offer for crucial machine learning problems such as transfer or compositional modeling. Our systems sheds light on these issues. Independent modules as sub-components could be learned using multiple domains or tasks, added subsequently, and transferred to other problems. This may constitute a step towards causally motivated life-long learning.

References

Aljundi, R., Chakravarty, P., and Tuytelaars, T. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3722–3731, 2017.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv:1511.07289*, 2015.

Daniušis, P., Janzing, D., Mooij, J., Zscheischler, J., Steudel, B., Zhang, K., and Schölkopf, B. Inferring deterministic causal relations. In Grünwald, P. and Spirtes, P. (eds.), *26th Conference on Uncertainty in Artificial Intelligence*, pp. 143–150, Corvallis, OR, 2010. AUAI Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Hyvarinen, A. and Morioka, H. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pp. 3765–3773, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Janzing, D. and Schölkopf, B. Causal inference using the algorithmic Markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2): 181–214, 1994.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lee, S., Purushwalkam Shiva Prakash, S., Cogswell, M., Ranjan, V., Crandall, D., and Batra, D. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in Neural Information Processing Systems 29*, pp. 2119–2127, 2016.

Pearl, J. *Causality*. Cambridge University Press, 2000.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of Causal Inference*. MIT Press, 2017.

Rojas-Carulla, M., Schölkopf, B., Turner, R., and Peters, J. Causal transfer in machine learning. *arXiv:1507.05333*, 2015.

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. M. On causal and anticausal learning. In Langford, J. and Pineau, J. (eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1255–1262, New York, NY, USA, 2012. Omnipress.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*, 2017.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 4, 2017.

Zhang, K., Zhang, J., and Schölkopf, B. Distinguishing cause from effect based on exogeneity. In *Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 261–271, 2015.

A. Additional results.

A.1. Too many or too few experts.

Too many experts. When there are too many experts, for most tasks only one wins all the examples, as shown in Figure 9 where the model has 16 experts for 10 tasks. In this case the remaining experts do not specialize at all and therefore can be removed from the architecture. Had several experts specialized on the same task, they could be combined after determining that they perform the same task. Since the accuracy on the transformed data tested on the pretrained classifier reaches again the upperbound of the untransformed data, and since the progress is very similar to that illustrated in Figure 6, we omit this plot.

Too few experts. For a committee of 6 experts, the networks do not reconstruct properly most of the digits, which is reflected by an overall low objective function value on the data. Also, the accuracy achieved by the pretrained MNIST classifier does not exceed 72%. A few experts are inevitably assigned to multiple tasks, and by looking at Figure 9 it is interesting to see that the clustering result is still meaningful (e.g., expert 5 is assigned to left, down-left, and up-left translation).

B. Details of neural networks

In Table 1 we report the configuration of the neural networks used in these experiments.

For the approximate identity initialization we train each network for a maximum of 500 iterations, or until the mean squared error of the reconstructed images is below 0.002.

C. Transformations

In our experiments we use the following transformations

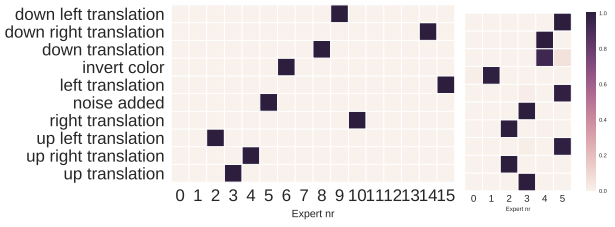


Figure 9. The proportion of data won by each expert for each transformation on the digits from the test set, for the case of 10 mechanisms and more experts (16 on left) or too few (6 on the right). Note how on the left experts 0, 1, 7, 11, 12, 13, do not win any data points, and can therefore be discarded.

- Translations: the image is shifted by 4 pixels in one of the eight directions up, down, left, right and the four diagonals.
- Contrast (or color) inversion: the value of each pixel — originally in the range $[0, 1]$ — is recomputed as $1-$ the original value.
- Noise addition: random Gaussian noise with zero mean and variance 0.25 is added to the original image, which is then clamped again to the $[0, 1]$ interval.

Table 1. Architectures of the neural networks used in the experiment section. BN stands for Batch normalization, FC for fully connected. All convolutions are preceded by a 1 pixel zero padding.

Expert
Layers
3×3, 32, BN, ELU
3×3, 32, BN, ELU
3×3, 32, BN, ELU
3×3, 32, BN, ELU
3×3, 1, sigmoid

Discriminator
Layers
3×3, 16, ELU
3×3, 16, ELU
3×3, 16, ELU
2×2, avg pooling
3×3, 32, ELU
3×3, 32, ELU
2×2, avg pooling
3×3, 64, ELU
3×3, 64, ELU
2×2, avg pooling
1024, FC, ELU
1, FC, sigmoid

D. Notes on the Formalization of Independence of Mechanisms

In this section we briefly discuss the notion of independence of mechanisms as in Janzing & Schölkopf (2010), where the independence principle is formalized in terms of algorithmic complexity (also known as Kolmogorov complexity). We summarize the main points needed in the present context. We parametrize each *mechanism* by a bit string x. The Kolmogorov complexity $K(x)$ of x is the length of the shortest program generating x on an a priori chosen universal Turing machine. The **algorithmic mutual information** can be defined as $I(x : y) := K(x) + K(y) - K(x, y)$, and it can be shown to equal

$$I(x : y) = K(y) - K(y|x^*), \tag{4}$$

where for technical reasons we need to work with x^*, the shortest description of x (which is in general uncomputable). Here, the conditional Kolmogorov complexity $K(y|x)$ is defined as the length of the shortest program that generates y from x. The algorithmic mutual information measures the algorithmic information two objects have in common. We

define two mechanisms to be **(algorithmically) independent** whenever the length of the shortest description of the two bit strings together is not shorter than the sum of the shortest individual descriptions (note it cannot be longer), i.e., if their algorithmic mutual information vanishes.[6] In view of (4), this means that

$$K(y) = K(y|x^*). \tag{5}$$

We will say that two mechanisms x and y are independent whenever the complexity of the conditional mechanism $y|x$ is comparable to the complexity of the unconditional one y. If, in contrast, the two mechanisms were closely related, then we would expect that we can mimic one of the mechanisms by applying the other one followed by a low complexity conditional mechanism.

[6] All statements are valid up to additive constants, linked to the choice of a Turing machine which produces the object (bit string) when given its compression as an input. For details, see Janzing & Schölkopf (2010).