Expanding the Domain of Learned Independent Causal Mechanisms

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Previous work has shown that a GAN model architecture can be utilized to reconstruct a canonical distribution from a set of independently transformed distributions [1]. This project implements the GAN architecture with the original set of transformations to create a baseline to compare to the original paper. The same architecture is then utilized with a new domain of transformations and compared to the baseline results to determine how the process generalizes to other transformations. While the experimental results were found to not match the original paper's results, a comparison and analysis between the baseline domain and new domain is done to evaluate the relative effectiveness of the architecture with a different set of transformations.

1 Introduction

It is not always practical to train a specialized model for every required task, either due to the number of tasks or from lack of training data. As such, the idea of domain adaptation has become a popular field of research. This is driven by the fact that a model that is capable of generalizing across multiple tasks is highly desirable.

A prior paper by Parascandolo et al. [1], explores the idea of using causality with a mixture of experts in order to learn a model capable of adapting to new domains of noisy data. They note that humans are capable of automatically recognizing distorted or noisy symbols without having to be told what the original symbol was. They argue that humans achieve this by using some sort of mechanism in their brain, that can generalize independently of the input domain.

$$p(\mathbf{x}) = p(x_1, ...x_d) = \prod_{j=1}^{d} p(x_j | \mathrm{pa}_G^j) \tag{1}$$

Using the independent mechanism assumption [1, 2], the authors then construct a joint Markovian density with respect to a causal graph, as shown in (1). Then this joint density can be considered as a generative process, that generates x_j from its parents pa_G^j. This can be further related back to causality [3], as the mechanism that generates x_j is analogous to a noise variable that can be found in structural equation models.

This project aims to expand on the original paper [1], by applying the model architecture and training method on a new domain of transformations. The original transformations encompassed translations, additive noise and pixel inversion, whereas in this project the the transformation domain is expanded to include rotations, stretches and compressions. A baseline model is trained on the original transformations in order to compare the results from the new transformations.

2 Related Works

As this project is an extension of the *Learning Independent Causal Mechanisms* paper, it is related to work done in the fields of mixtures of experts, domain adaptation and causality.

In terms of mixtures of experts, Shazeer et al. [4], created a new layer architecture that consists of 1000 expert sub-networks, and learn a gating network that selects the best combination of experts for a given sample. However these networks are simple feed-forward networks, and are not generative like the experts in used in this project. Lee et al. [5] uses a stochastic learning method to train only the best performing model on a given sample, similarly to how the experts are trained in this project. The model performance in Stochastic Multiple Choice Learning is not learned jointly with the experts however, and is just a static error calculation. Aljundi et al. [6] use an auto-encoder as a gating mechanism to select which expert to train for a given task sample. Their gating method relies on using past performance on related tasks however, which means they do not utilize independent experts and mechanisms.

Bousmalis et al. [7] use an unsupervised adversarial approach that learns a pixel-level transformation between two domains. This is done with only a single generator. Relating to both domain adaptation and causality, Benigo et al. [8] use the independent mechanism assumption to learn a structural causal model to perform transfer-learning between two distributions. Similarly to this project, they aim to de-rotate an image, but their process focuses more on learning a bivariate SCM, and does not learn a selection system for multiple domains. As mentioned earlier, this project draws on ideas from Pearl [3], as the learned experts in this project are an alternative representation of the structural causal models used in causality, and the aforementioned mechanisms are the noise variables imposed on the system.

3 Proposed Method

The overall problem can be formulated as such [1]; given a true distribution P, and a mixture of distributions Q, there are some number of independent mechanisms $M_1, ...M_N$ that map $P \to Q$. I.e. for each M_i and $\exists p \in P$, $M_i(p) = q_i \in Q_i$. Then by the IM assumption, these Q_i's will be independent from each other. Then, want to learn a group of experts $E_1, ..E_N$ such that an expert E_i is an inverse mapping of M_i, so that we can recover p from q_i, i.e. $E_i(q_i) = p$.

Parascandolo et al. [1] also go into more detail, and formalize the independence of the mechanisms as outlined in Janzing et al. [9]. They use Kolmogorov complexity [10] and bit strings x and y to define algorithmic mutual information:

$$I(x : y) = K(x) + K(y) - K(x, y) \tag{2}$$

These bit strings x and y are said to be parameterized representations of two different mechanisms, and $K(x)$ is the length of the shortest program capable of generating the bit string x. Using the shortest representation of x, x^*, the algorithmic mutual information can then be formulated as

$$I(x : y) = K(y) - K(y|x^*) \tag{3}$$

From this, two mechanisms can be declared algorithmically independent of each other if $I(x : y) = 0$, which from (3) implies that

$$K(y) = K(y|x^*) \tag{4}$$

i.e. that conditionally x has no effect on y, and are independent of each other. Therefore it can be seen that since the output of a mechanism y is independent of the output for mechanism x, the transformed distributions $Q_i, ..., Q_N$ are independent of each other and the IM assumption holds.

3.1 Transformations as Mechanisms

Parascandolo et al. [1] chose to represent the aforementioned mechanisms as various physical transformations. When applied to some canonical distribution, they result in several distinct sets of transformed distributions. Examples of the transformations used can be seen in figure (1). These transformations include translations, Gaussian noise addition and colour inversion. This project

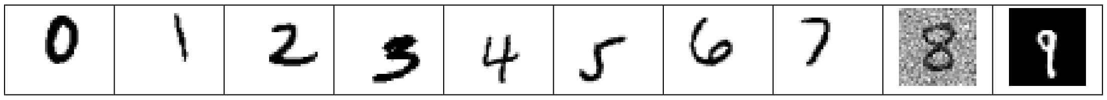

Figure 1: Examples of transformations used in the original paper by Parascandolo et al. [1]. The first eight images (0-7) show the various translations, 8 shows added noise and 9 shows the pixel inversion.

aims to expand the domain of transformations that is applicable to the proposed architecture. As many of the previously used transformations are simple affine transformations, like the translations, this paper explores the use of more complex affine transformations. These include rotations, as well as stretching or compressing both vertically and horizontally.

More detail on how these transformations are applied to the dataset is discussed in section (4).

3.2 Model Architecture

The model used is a Generative Adversarial Network (GAN) [11], consisting of a single discriminator and N experts. The discriminator architecture can be seen in fig. (3), and the expert architecture can be seen in fig. (4).

3.3 Training Method

This project follows the same training method as outlined in the original paper [1]. First, N experts and a single discriminator are initialized, where N is the number of transformations tin the domain. Optimizers were initialized for each expert and discriminator as well. Then, the experts were pre-trained using a method called Approximate Identity Initialization (AII) [1], which is shown in algorithm (1), where MSE is the mean square error.

Algorithm 1: Approximate Identity Initialization

```
1 for $t = 0, 1, \ldots, 500$ do
2     $x$ = sample_transformed_image()      // Get a random transformed sample image
3     for $E_i$ in $UE$ (Uninitialized Experts) do
4         $y_i = E_i(x)$                    // Get output for $i^{th}$ expert
5         if $MSE(x, y_i) < 0.02$ then
6             $UE \leftarrow UE - \{E_i\}$   // Remove $E_i$ from uninitialized set
7         else
8             $E_i \leftarrow E_i - \nabla MSE(x, y_i)$   // Update $E_i$ with gradient step
```

Once the AII training is complete, the adversarial training process begins. In each iteration a minibatch of clean and transformed images are sampled. Each transformed image is inputted into each expert E_i, and then the discriminator is updated using the clean images as well as the outputs of

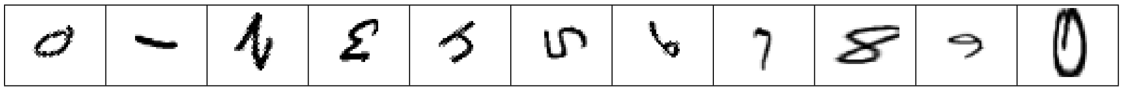

Figure 2: Examples of an expanded domain of transformations used in this project.

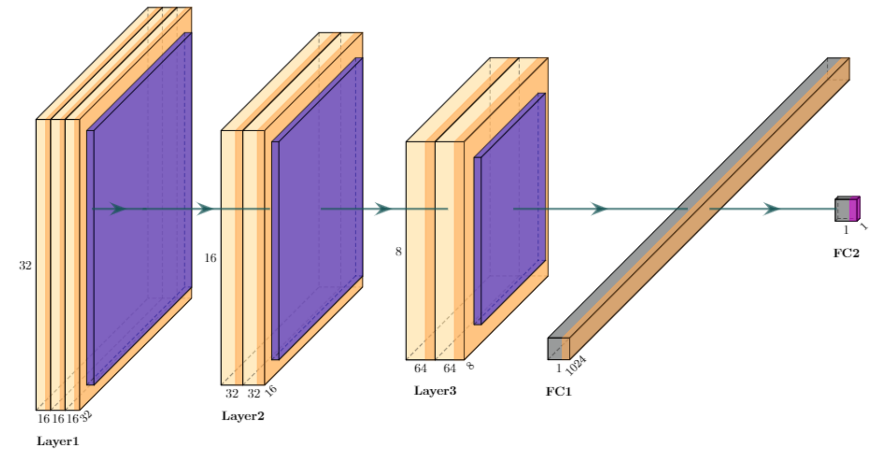

Figure 3: Architecture of the discriminator. Yellow blocks represent convolution layers, orange blocks represent an ELU activation, purple blocks represent an average pooling, gray blocks represent a fully connected layer and the pink block represents a sigmoid activation. Each convolutional filter is 3×3, and the pooling filters are 2×2. The first layer consists of 16 channels, the second 32 channels and the third 64 channels.

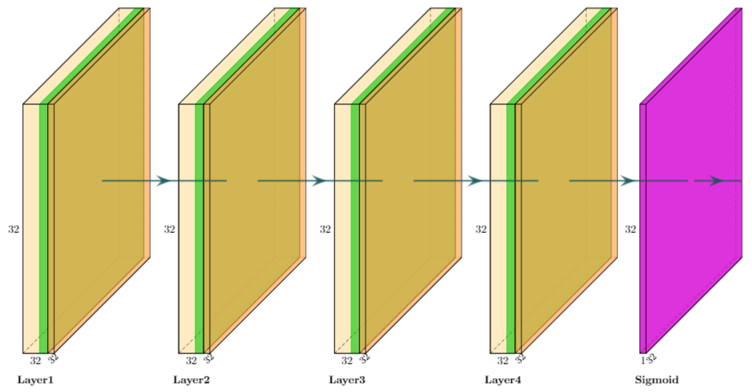

Figure 4: Architecture of an expert. Yellow blocks represent convolution layers, green blocks represents a batch normalization, orange blocks represent an ELU activation, and the pink block represents a sigmoid activation. Each convolutional filter is 3×3, and each convolutional layer consists of 32 channels.

the experts. The experts that achieved the highest discriminator score on a transformed sample are updated on the respective samples that they scored highest on.

Algorithm 2: Adversarial Training Process

```
for $t = 0, 1, \ldots, maxiter$ do
    $x, x' =$ sample_minibatch($size$)            // Sample a random minibatch
    for $E_i$ in $Experts$ do
        $y'_i = E_i(x')$                           // Get output for $i^{th}$ expert
        $c_i = D(y_i)$                 // Get discriminator scores for $i^{th}$ expert
    $d_{loss} = BCE(x, 1) + BCE(y', 0)$
    $D \leftarrow D + \nabla d_{loss}$                      // Update discriminator
    $E_i \leftarrow E_i - \nabla \operatorname{argmax}_i BCE(c_i, 1)$        // Update winning experts
```

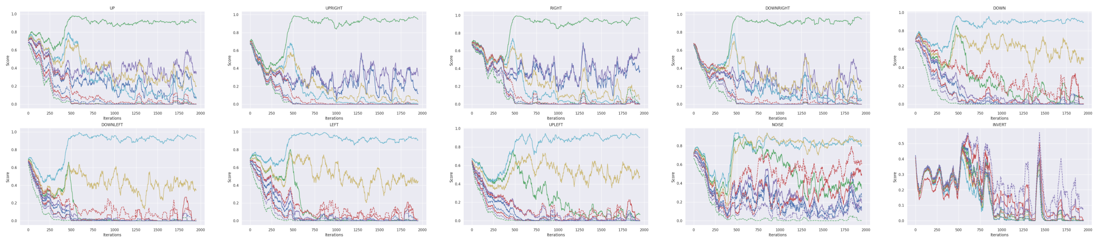

Figure 5: Score plot for each transformation used in the original domain. Each line refers to a different expert. All scores were smoothed using a running average over a window of 50 iterations.

This training process can be seen in algorithm (2). Where BCE is the binary cross entropy loss (equation (5)), where x is a score vector outputted by the discriminator and y is the label vector denoting real or transformed samples.

$$BCE(x, y) = y\log(x) + (1 - y)\log(1 - x) \tag{5}$$

4 Experiments

4.1 Transforming the Dataset

The MNIST digit recognition dataset was used for both sets of transformations. The dataset consists of 60000 training images, and 10000 test images. Since the process is unsupervised, only the training dataset is used. The training set is partitioned into two groups of 30000, with the first 30000 images being used for the transformations and the second group of 30000 is left unchanged. Parascandolo et al. [1] did not detail their process for transforming the dataset, so a cyclic process was used. When generating the transformed images, the image partition set aside for transforming is first randomly permuted. Then the permuted set is iterated over, and the transformations are applied cyclically in order to ensure each transformation will show up in the final transformed dataset in equal proportion.

The specific details for the original translations are as follows: translations were by 4 pixels in all directions, the Gaussian noise had a mean of 0 and a variance of 0.25, and each inverted pixel p' in an inverted image is found using $p' = 1 - p$ [1]. For the transformations in the expanded domain, rotations were done in increasing increments of $45°$ up to $315°$, and all stretches and compressions use a factor of 1.75.

4.2 Baseline Results

A baseline aiming to replicate the results of the work done by Parascandolo et al. [1] was done in order to be able to compare the results of the new transformations. The same model architecture, default optimizer parameters (ADAM) and minibatch size (32) were utilized. Unfortunately, this project was not able to replicate the results found in the original paper. See figure (5) for the score plots across all experts, and figure (6) for examples of the inverted transformations learned by the experts. The figures display the (relative) best results from the 10 runs that were conducted.

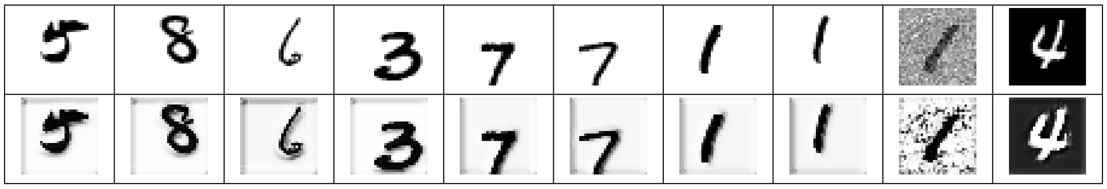

Figure 6: Examples input images of each type of transformation (first row), and the corresponding output of the expert that scored the highest for each transformation after the training finished (second row).

As it can be seen, only a few experts specialized in transformations, and the ones that did began specializing over multiple transformations. These experts tended to specialize over translations with overlapping translation directions. E.g. as seen in fig. (5), the two experts that specialized over the translation transformations were for the directions UP/RIGHT and DOWN/LEFT respectively. Several experts started to excel in the de-noising transformation, with no singular expert out performing out all the others. In this specific run none of the experts began to specialize in the inversion transformation, but in some other runs a single expert would begin to specialize with a similar output as can be seen in fig. (6).

A number of techniques were employed in an attempt to stabilize the training procedure, including the use of Wasserstein loss [12], two time-scale update rule (TTUR) [13], and one-sided label smoothing [14]. In all cases results were comparable to the base results shown above, an example of which can be seen in figure (7).

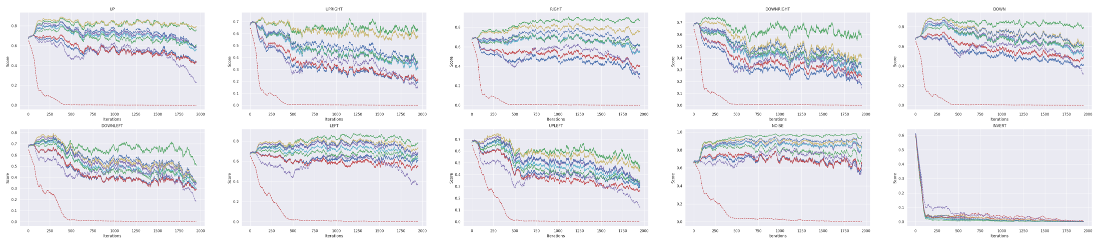

Figure 7: Score plots of experts when using Wasserstein distance as a loss function. A lower score indicates an expert's output is closer to the canonical distribution.

Possible reasoning behind why the original results could not be replicated, as well as further analysis of the baseline results, is discussed in section (5).

4.3 Expanded Results

Similarly to the baseline experiment, the expanded domain of transformations did not result in the expected outcome. In every training run only one expert began to specialize, and would end up winning samples for all of the transformations. See figure (8) for the score plots and figure (9) for examples of the generated outputs.

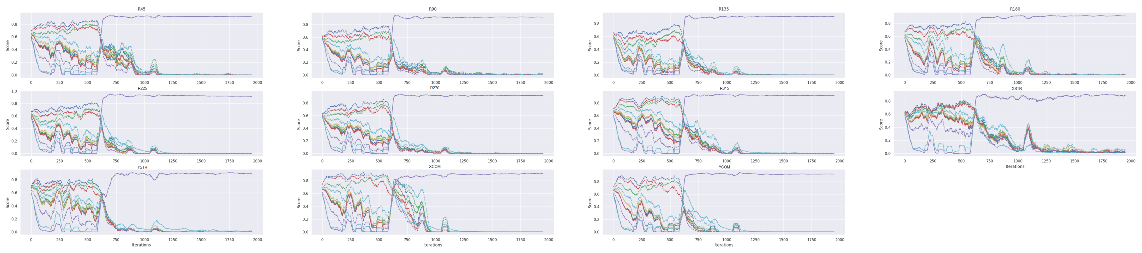

Figure 8: Score plot for each transformation used in the expanded domain. Each line refers to a different expert. All scores were smoothed using a running average over a window of 50 iterations.

The GAN stabilization techniques described in the previous section were used, and again it was found that they did not improve the results of the model. Further discussion and comparison to the baseline results can be found in section (5).

5 Discussion

5.1 Possible Reasoning for Incorrect Results

The first potential reason that the experimental results of the baseline do not match those of Parascandolo et al. [1], is that there is an implementation error. This project attempted to follow the outlined training process found in Algorithm 1 [1] from the original paper as closely as possible, but the proposed algorithm was relatively sparse in details. There could have been intermediate steps

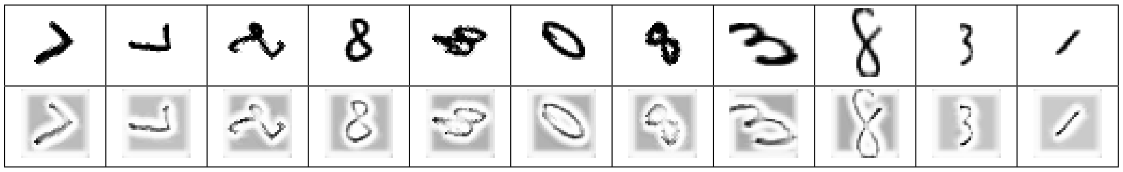

Figure 9: Examples input images of each type of expanded transformation (first row), and the corresponding output of the expert that scored the highest for each transformation after the training finished (second row).

omitted from the shown algorithm that affected the training process, or there could be a hidden bug in the code for this project. The loss function as seen in the original paper (equation (3) [1]), is binary cross entropy. In this project's implementation the Pytorch BCELoss function is used, however the original paper might have used a custom loss that differed from the Pytorch implementation. Unfortunately, the authors have neglected to upload a publicly available repository of their code, so a direct code comparison is not possible.

Another potential reason for the difference in results is from the generation of the training dataset. As mentioned before, Parascandolo et al. [1] do not disclose the method they use to transform the original digit images. It is assumed they do so randomly, with a relatively even proportion of each type of transformation. Under these assumptions, the aforementioned cyclic process is used. However, it could be possible that some other process was used that resulted in a better dataset for learning the inverse transformations, although theoretically it should not make a difference.

A third reason could be the fact that the original paper states that their implementation was done in parallel [1], while this project was implemented sequentially. While the experts should theoretically specialize in both settings, it could be that training the experts across multiple processes helped enforce the independence between them.

Of the listed reasons, it is most probable that the reproducibility issue is due to the first reason, as the second and third reasons should theoretically not have much effect on the outcome.

5.2 Analysis and Comparison of Results

Assuming that the inability to reproduce the original paper's results lies in an implementation error or missing training step, the results of the expanded domain can still be compared to the baseline domain.

When observing the baseline results in fig. (6), it can be seen that the experts that begin to specialize do so on groups of transformations that share a common direction. I.e. the first expert scored best on the images that had been shifted up, while the second image scored best on the images that were shifted down. In both cases it can be seen that the experts began to start shifting the digits back toward the center, as evidenced by the gray pixels that were generated along the bottom or top of the digits respectively. In terms of the noisy images, most experts scored well, and the de-noised image is significantly less noisy than that of the original. Similarly, even though most experts scored low on the inverted images, it can be seen in the best output that the black pixels have begun to get lighter, while the white pixels begin to get slightly darker. So in all transformation cases the experts have begun to learn the inverse mechanism transformation, but were too weak to properly reconstruct the original sample. Using the scores from fig. (5) in conjunction with this, it seems clear that a major issue in the implementation is that the discriminator is failing to learn a proper representation of the canonical distribution. This results in the experts scoring higher than they should, and as such they fail to improve their ability to reverse a given mechanism transformation. As mentioned earlier, TTUR [13] was used in an attempt to remedy this by increasing the learning rate of the discriminator relative to the experts, but the problem continued to persist.

As seen in the results in fig. (8), when running the model on the new domain the experts fail to specialize at all, with only one expert winning out on all transformations. This result suggests that the experts were unable to meaningfully differentiate the differences between the transformations. All the transformed images in the expanded domain (fig. (9)) were still centered, while the experts in the baseline experiments were able to differentiate some transformations based off a vertical offset from the center. Since only one expert began to specialize across all transformations, it also failed to begin to learn any of the inverse transformations, unlike the experts in the baseline. Again, this seems to be the result of a weak discriminator failing to learn the true distribution. Furthermore, it was found that using the expanded domain of transformations resulted in greater model instability due to the vanishing gradient problem, which was not prevalent when using the original transformations.

While the baseline results are not nearly as conclusive as those in the original paper, it can be seen that there was some partial reproduction of the original results. Unfortunately the results of the expanded domain are inconclusive, as there does not seem to be any potential of reproducing results in line with those of the original paper. The original paper [1] claims that the default learning rate for the ADAM optimizer (1×10^{-4}) was used for both the expert and the discriminator optimizers, and in their score plots the specialization convergence generally tended to happen before 250 training iterations. This seems to imply some intermediate step or hyper-parameter that was not reported in the original paper is causing the disparity in experimental results. Considering the performance of the baseline transformations compared to the expanded domain of transformations, it seems unlikely that using Parascandolo et al.'s [1] implementation on the expanded domain would yield promising results.

This calls into question the validity of the IM assumption made by Parascandolo et al. [1]. While their results do seem to show that there is merit in the use of GANs for this form of domain adaptation, there still seems to be some overlap between the transformations they used. Is a transformation that translates a digit vertically and horizontally to the left actually completely independent from a transformation that translates a digit vertically and horizontally to the right? Both contain a vertical translation component, and as seen by the baseline results in fig. (6), the experts group these vertical translations together, so they must find that there is some overlap between the transformations. This implies that the mechanisms used to transform the images are not necessarily completely independent of each other. Therefore the IM assumption made is likely not an absolute requirement for the proposed architecture to work, and the relative independence of the transformations is more like an indicator of how effective the method would be. I.e. while the translation transformations have a degree of dependence on each other, they are still independent from the noise or inversion transformations, so the method still performs well. Therefore, while the results found by Parascandolo et al. [1] are definitely of merit in the fields of domain adaptation and mixtures of experts, its connection to the field of causality is ultimately questionable.

6 Summary

A set of experiments were run in order to test and expand upon the results found by Parascandolo et al. [1]. A baseline experiment using the domain of transformations in the original paper was done for use in analysis. Experiments were also run using the same model architecture and parameter settings on an expanded domain of transformations involving rotations, stretches and compressions. The results of the original paper were only able to be partially replicated by the baseline experiments. The results of the expanded domain were inconclusive, but when compared to the baseline implied that the expanded domain would not be feasible if used on the original implementation. Furthermore, the results of the baseline experiments suggested that the IM assumption made by Parascandolo et al. [1] is unnecessary as a strict requirement for the proposed method to work.

In the future, further avenues can be explored in order to further tie the method proposed by Parascandolo et al. [1] into the field of causality, as it seems that the method is only tenuously connected. Other potential work can be done in attempting to use the proposed architecture on a more complex image dataset. Since this project and the original paper only used the grayscale MNIST dataset, it would be interesting to see how the architecture performs on RGB images. Another dataset that was originally planned to be explored in this project before running into implementation issues was testing the model on a CAPTCHA dataset, where there are multiple transformations and multiple digits per image, and in doing so work towards an expert selection process that resulted in the optimal combinations of experts for a given CAPTCHA image.

References

[1] Giambattista Parascandolo, Mateo Rojas-Carulla, Niki Kilbertus, and Bernhard Schölkopf. "Learning Independent Causal Mechanisms". *CoRR*, vol. abs/1712.00961 (2017). arXiv: `1712.00961`.

[2] Bernhard Schoelkopf et al. *On Causal and Anticausal Learning*. 2012. arXiv: `1206.6471 [cs.LG]`.

[3] Judea Pearl. *Causality: Models, Reasoning, and Inference*. 2nd. Cambridge University Press, 2009.

[4] Noam Shazeer et al. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". *CoRR*, vol. abs/1701.06538 (2017). arXiv: `1701.06538`.

[5] Stefan Lee et al. "Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles". *CoRR*, vol. abs/1606.07839 (2016). arXiv: `1606.07839`.

[6] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. "Expert Gate: Lifelong Learning with a Network of Experts". *CoRR*, vol. abs/1611.06194 (2016). arXiv: `1611.06194`.

[7] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks". *CoRR*, vol. abs/1612.05424 (2016). arXiv: `1612.05424`.

[8] Yoshua Bengio et al. "A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms". *CoRR*, vol. abs/1901.10912 (2019). arXiv: `1901.10912`.

[9] Dominik Janzing and Bernhard Schoelkopf. *Causal inference using the algorithmic Markov condition*. 2008. arXiv: `0804.3678 [math.ST]`.

[10] A.N. Kolmogorov. "On Tables of Random Numbers". *Theoretical Computer Science.*, vol. 207, no. 2 (1998), pp. 387–395.

[11] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]`.

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: `1701.07875 [stat.ML]`.

[13] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium". *CoRR*, vol. abs/1706.08500 (2017). arXiv: `1706.08500`.

[14] Tim Salimans et al. "Improved Techniques for Training GANs". *CoRR*, vol. abs/1606.03498 (2016). arXiv: `1606.03498`.