NLDL Winter School 2023

Candidate 25

1. Introduction

This is a report from the 2023 Northern Lights Deep Learning (NLDL) conference. The first four pages contains a summary from each of the topics covered Monday 9th of January and Friday 13th January. I have primarily used the presentations as a source, but supplemented a little with papers I have found online. Citations to these sources are given. I have omitted some content from all the presentations to adhere to the length requirements. A detailed programme can be found on the website: https://www.nldl.org/winter-school.

2. A gentle introduction to Deep Reinforcement Learning

In Reinforcement Learning (RL), one or more agents interact with a given environment and teach themselves behaviours that maximize a goal-function. Often this relies a great number of iterations; e.g. AlphaGo Zero played 4.9 million rounds of Go with itself before being released [30]. This got news coverage all over the world as Go had long been considered to be a game too complicated for computers.

RL agents operates in discrete time. At each timestep t the agent observes its environment. Each observation O_t modifies the agent state s_t, a processed representation to the history of events. Based on s_t the agent chooses it's action A_t. The choice is made by choosing the action thought to maximize the reward R_t of the agent in the future (next state). The expected return from the eyes of an agent is measured by it's goal function G_t, in simple form being the sum $G_t = \sum_{k=t+1}^{\inf} R_k$, i.e. a sum of all rewards seen collected from a series of actions. A problem with this simple goal function is that the agent can get stuck. E.g. in a dead end of a labyrinth where the goal function is trying to minimize the distance to the exit. This can lead a robot into a dead end, refusing to take a step back to find another path to the goal. One way to work around these situations is to introduce a discount factor, noted γ. It is used to ensure that rewards are less valuable over time. The goal function will then be modelled as $G_t = \sum_{k=t+1}^{\inf} \gamma^k R_k$ (which can also be expressed a recursive $R_{t+1} + \gamma G_{t+1}$). Hence, if you stick in a dead end for too long, it doesn't pay off, and any other path that penultimately gets to another reward faster is better. The use of discount factors in reward functions is referred to as *Temporal Difference (TD) learning*.

In RL it is also common to discuss *value functions*. This is the expected value from being in a state s: $V_\pi = \mathbb{E}_\pi[G_t|s = s_t]$. π is here defined as the policy for what action to choose. Usually we cannot know exactly what the outcome of an action is, therefore we model actions by stochastic policies $\pi(a|s)$ which gives likelyhood of future actions. An example is a robot taking a step forward, but sliding on a banana peel and consequently falling. This lesson means that re-visiting the same state later in the future should not yield the same action, even if the policy is the same. The same does not apply to e.g. chess, where we can set a deterministic policy since we know exact outcomes from actions (moving pieces). Analogous to the value function is the *action value*: $q_\pi = \mathbb{E}[G_t|S_t = s \cap A_t = a]$. This function (as given here) is strict to the policy π, i.e., in order to predict the future value of actions, you have to stay with the same π. The q is often referred to as *quality*.

2.1. Deep RL

Keeping a mapping between states and their expected values can be exhaustive for computer memory. Finding the right policy can also be difficult. If we think of neural network for what they are, *function approximators*, it seems intuitive to use them for approximating the value function and/or action policy. This helps us resolve the problem of finite available space in action-value mappings. Hence, the field of **Deep Reinforcement Learning (DRL)** emerges. There are several examples already of popular tools/models built with DRL, such as AlphaTensor [37] for matrix multiplication, DeepNash [24] for the game of Stratego, etc.

The neural nets in DRL (a.k.a Deep Q Network (DQN)) lacks ground truth, but rather rely on rewards for optimization. The problem is that these rewards may come at delayed moments or at the influence of unknown variables. This is essentially noise in the learning loop. Therefore, there are several techniques used in conjuction with the neural nets to improve their accuracy. One such trick is experience replay: steps are recorded with (s_t, a_t, s_{t+1}, r_t) in a replay buffer. Now, whenever we perform an action, we can learn not only from the change in environment, but we may use extra samples from past events.

2.2. Environment Learning

So far in this summary the RL approaches have been *model-free*: to learn of the world, actions are performed and evaluated. The alternative is *model-based* RL, wherein the agent can plan ahead by making predictions about how actions affect the world: $p(s_{t+1}|s_t, a_t)$. The advantage of model-based RL is primarily it's sample efficiency since we can prune away more actions during the learning phase. This is the case of Alpha Zero [29]. There have, however, been other strides in e.g. MuZero [28], which can also train its own model to be used by RL to predict future actions.

3. Self-Supervised Learning: Training Targets and Loss Functions

There are now four different training paradigms in machine learning. 1) is supervised learning, in which an annotator labels data to be used by a machine learning model. 2) is semi-supervised, which can combine unlabeled data and labelled data together. 3) is unsupervised learning, in which no input is given to the model except raw data. Approach 1 and 2 both depend on annotations which may be costly (and tedious) to aquire and approach 3 is fast but has limited learning capacity. Self-Supervised Learning (SSL) is the fourth paradigm intending to resolve the headaches from the former 3. SSL can perform unsupervised learning as a *pretext* task that can then be used downstream either as a stand-alone feature predictor or to be used in conjuction with labelled data later on. SSL has made big advancements in Natural Language Processing (NLP), for example in the most recent ChatGPT (made by OpenAI [22]) (albeit its underlying model based on GPT-3 also uses other techniques [4]).

3.1. Encoding Context

For speech and text, one technique to make SSL efficient is Autoregressive Predictive Coding (APC) [8]. This technique relies on encoding context in the form of past elements into a latent space (a model's interpretation) in order to predict future elements. This prediction is made using a softmax-layer which effectively calculates the probability of future tokens (referred to as frames). The loss function of the model also uses the probability of past and future frames as an optimizer. The difficulty of this approach is having enough computer memory to store all the past tokens. To aid this it is helpful to use *transformers*, which have attention-based learning and may compress some of past frames. This could e.g. be elimiting noise in audio sounds.

3.2. Transferrability

Often, to make good use of SSL, you have to optimize for the *context*. For example, data2vec [**?**], another SSL technique for NLP, relies on having contextualized training targets to learn the embedding of words and documents. I.e. by calculating a term frequency-inverse document frequency (tf-idf) from carefully selected documents it is able to calculate a good word embedding to be used in later analysis. For non-document related domains (such as images) the tf-idf would be off, however. Thus, SSL does not resolve a big issue in modern day Artificial Intelligence (AI): lack of transferability of models.

4. The Challenge of Unverfiability in eXplainable AI Evaluation

Deep Learning (DL) depends on several hidden layers that can have millions or billions of neurons. These neural nets are therefore often regarded as a "black box" because it is difficult to get an overview of how they make their predictions. Understanding how a model works may be important, especially if the model is going to be used in critical settings such as medicine or any kind of hazard detection.

Explainable AI (XAI) aims to resolve these issues. XAI not only increases our trust in how AI models work, they can unravel biases and generate new insights. If you for example want an AI to separate dogs and wolves, it might be that the AI is learning to recognize the background setting, which for wolves will more often be outside in the woods. It might also be that a model picks up artifacts in photos not seen or known to humans that are nevertheless relevant and could lead to new insights (for humans).

One example technique for *explainability* is done by occluding different parts of inputs to an DL model. If the model is still able to accurately predict the classification, that means that the relevant features are still in the photo. You can repeat this process to learn the essential features that the model depends on. One example technique for *comprehension* is mapping model outputs to a three-dimensional or two-dimensional space (also covered in Section 8.2.1) to visualize how a model maps different features. If two (theoretically) similar inputs don't end up close to each other in 2D/3D space, this could mean that model is picking up unknown artifacts.

Methods like data occlusion is *model agnostic*: even if you modify or change the model the method would still operate in the same way. There is another family of techniques that are *model aware*. One such method is saliency mapping [31], a gradient-based method. Somewhat similar to data occlusion, input to a machine learning model is modified. This time, however, the output from the model is tracked to determine its gradient with each change. By tracking what regions impact the gradients the most, the most relevant regions are established. The downside with model-aware methods is that their dependency of the model, and if the model is unstable, the explanations may be unstable, too.

4.1. Verifying explanations

But, how robust are the metioned explanations? [1] finds that it is possible to modify input images in a way that is (barely) unoticable to humans but nevertheless dramatically modifies the output explanations. One such pertubation is shown in Figure 1. There has therefore been made several metrics for getting quantifiable verifications of how good an explanation is. Several of these are implemented in Quantus [16], a python API that lets you automatically get metrics for 6 different dimensions of explanations:

1. Faithfulness: to what extent does the explanation follow the model? e.g.: do two similar explanations have similar inputs?
2. Robustness: If you modify the input slighly, does the explanation remain the same?
3. Localisation: Does the explanation consider the entire input, or are there just core regions of interest that matters?
4. Complexity: can an explanation rely on only a few features, or is the entire input necessay?
5. Randomisation: If you modify the model hyperparameters, is the explanation accuracy impacted
6. Axiomatic: do explanations have axiomatic properties such as *completeness*, *non-sensitivity* and *input invariance*?

5. Data representativity and low-resource in Deep Learning

DL depends on many iterations with data to improve its accuracy. If there are biases in the training data, this will also affect the inference results. Therefore, having *representative samples* for model training is highly desirable. But, the term representative is subject to interpretation, sometimes leading to fallacious claims, even in research. It is also important to remember that we sometimes do not want our model to only work well in general cases. A medical scanner should work just as well for a representative user as any other user. E.g. most hearth disease occurs in elderly, but scanners should work well on children, too, unless you build them separate machines.

According to [18, 19] there are 6 notions that are used to claim representativity. The first is **assertive claims**, which state that they have general or all-covering data without explaining why. One might say that since ImageNet [12] has many photos, it *has to be* general, but this has been found to not be the case [33]. Second, there is **"the miniature"**, wherein sub-populations might be over- or undersampled. These miniature models may suffer from a poor choice of

Figure 1. Left side: original image with its explanation. Right side is a seemingly similar image, but it is modified in such a a way that the explanation output is completely different. The figure is taken from [1]

stratifying attributes, either from misconceptions or subjective biases of the sampling author. Third, there is **absence or presence of selective forces**. A survey will never measure non-responsders for example ("selection force") and a random sample may not be representative if e.g. the sampling phase/time is chosen poorly ("absence of selective forces"). Forth, there is the **Typical/Ideal** sample, wherein samples used are supposedly averages of their subgroup. One example is choosing cluster centers from Gaussian Mixture Models (GMM) (*details omitted*). Fifth, there is **Coverage Claims**. In a "Noah's Ark-esque" fashion, subgroups of data are sampled in equal numbers so as to avoid the potential of other biases. There are pitfalls here too, however, such as not accidentally undersampling and thus not covering the entire data span. To do full coverage, it is often common to see both random and non-random sampling used together: selecting groups deliberately, but samples from that group randomly. Sixth, there is a the deferral of defining representativity, giving instead a **reference to sampling methods**. The notion is carried by the idea that all datasets will *always* be only a minitature model and subject to bias, so rather than claiming a *definitive* data representativity, you leave it to the reader.

[9] additionally defines two other notions particlarly relevant for AI. The seventh notion is **the copycat**, where synthetic data is used to claim representativeness. Finally, the last notion is the simplest one: you do not talk about representativity at all.

Assertive (first) and "no-notion" (eight) claims are to be

avoided in science. The rest may, however, be viable to use if you carefully assess the context. In any event, thoroughly outlining selection methods in any scientific work is always recommended.

6. High Performance Computing for Deep Learning

The Norwegian Research Infrastructure Services (NRIS) (formerly "the Metacenter") is a collaboration of multiple organisations in Norway to share experience and offer services and resources for research. One of these services are Sigma2, which offers a High Performance Computing (HPC) environment.

HPC are comprised of a large number of storage and processing units, allowing programmers to solve problems that are too big for consumer-level computers. Unlike another option, cloud environments, HPC store all their resources in the same location and usually have shared file drives. This makes it fast to launch distributed workloads.

6.1. Synchronization schemes

In DL, an immediate advantage of HPC is the higher number of Graphical Processing Unit (GPU) available, potentially removing the need for batch-processing. However, working with multiple GPUs introduces some challenges: how and when do we communicate between nodes? There's two primary modes: synchronous and asynchronous. Synchronous processing is often faster [17], but may be bottlenecked by slower or interrupted machines. Furthermore, synchronization points may be costly (slow), leaving nodes idling. Asynchronous processing mitigates these problems, but have other costs. Many asynchronous frameworks will e.g. have smaller workloads per kernel, giving extra overhead to load balancers that have to distribute remaining work [23].

7. Representation learning and learning with few data

Labelling data takes time and is tedious for annotators. One solution is *transfer learning*, using another model for your own data. However, this approach has been shown to only help for low-level features, thus still leaving a need for labelling your own datasets, even if the original model was trained on a dataset as large as ImageNet [12]. Models trained on this network has also been shown to be biased for textures rather than shapes.

Another solution to the labelling problem is *representation learning*. Here, the model is able to generalize features from training. One way to do so is the use of autoencoders. These neural networks are made to reconstruct input data after they have been compressed or distorted. This way, they should in theory be good at understanding the features

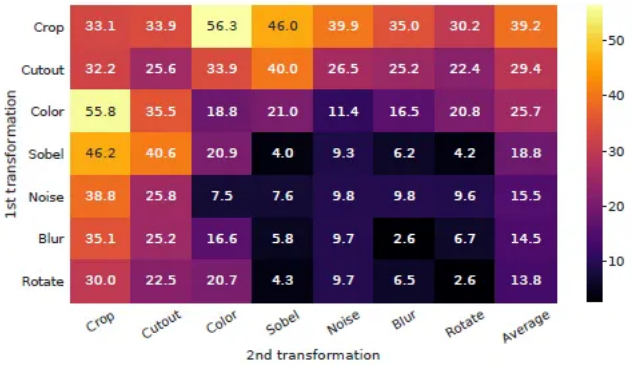


Figure 2. Figure taken from [6]. Each number corresponds to a linear evaluation of the SimCLR model using different augmentations. In the diagonal (except in average column) each transformation is only done once. For the rest there are two augmentations done together. Higher numbers represent more accurate results. The combination of crop and color have the highest evaluation scores for *for ImageNet*.

of their data. However, it is crucial that data is not biased or noisy, otherwise the learned features may be incorrect. SimCLR [6] has a nice table (reproduced in Figure 2) to show which alterations/augmentations work better or worse than others for their algorithm on ImageNet.

7.1. Contrastive Learning

Contrastive learning is a technique where you make a prediction both *positive* and *negative* versions on an input. The loss function will in effect try to shift the weights to assimilate the positive pair while pulling away the negative pairs inputs. One example of a loss function is using cosine similarity scores, where the output from each model is represented as a vector. The cosine similarity captures the angle between the vectors, and can thus be used as an indication of similarity.

The image alterations are based on *human priors*, meaning they work well for features that humans depend on like sizes, shapes or separating foreground and background. Thus they often work well on natural images. Though, for some domains such as medicine, they prove less efficient. E.g. in histopathology where there are many small details in each section of an image that are all important to make a classification. [7] suggests to instead use *data prior* for Histopathological images. One such example is using different levels of zoom as an image alteration. By zooming out, you *include* more data as input. The technique works nevertheless since you want the model to focus on an area of interest (such as a tumour) regardless of its surroundings.

8. Practical Project - Introduction

Lung Squamous Cell Carcinoma (LSCC) is a type of cancer with high recurrence rate and likelyhood of metastasis. There aren't many publicly available datasets with annotations available. One popular data source is The Cancer Genome Atlas (TCGA), which has image, clinical and genomic data available. We will be looking at H&E-stained biopsy slides, called Whole Slide Image (WSI).

[38] uses SSL to provide prognostic information of LSCC. They observe from other studies such as [25] that constrative learning tends to learn non-generalized features by using "shortcuts". One such shortcut might be to learn the background color of a an image to make a prediction rather than the features of each object. This can be prevalent in processing WSI that are used in histopathology. One reason is the inputs for the constrastive loss function do not find *negative pairs* from within the same slide. Thus, the features learnt are not feature-specific, but rather overfitted to each slide. They defend this claim by showing a single Uniform Manifold Approximation and Projection (UMAP) plot, reproduced here in Figure 3.

Inspired by the talk by Anna Hedström, I want to quantify the quality of the explanation mentioned in [38]. [20] points out that explanations are always *contextual*, i.e. that its quality will always depend on the what the reader wants to know and what the explanation agent has knowledge of. In my particular context, I am curious if the explanation could be useful at an early stage to indicate whether or not a model is overfitting or not. This is because I am currently writing a tool to help pathologists annotate datasets. In this case, there might be several annotations made in just for a few slidesIn this context, I want to provide continous feedback to the annotator to guide them. This can both decrease the time they need to annotate datasets and increase their trust in the model.

There are several possible pitfalls with a 2DUMAP map: you could adjust the sizes of the dots for each point to create an illusory feeling of more even distribution. Secondly, the figure uses the *assertive claim* that "the tiles from each slide are less clustered together". However, the figure only shows output from 8 slides which may have been chosen in a non-representative fashion, and since UMAP may be non-deterministic, the differences we see could be random, and without any metric axis on the plots, the "less clustered"-effect could be illusory.

Summarized: I've done the following work:

- Reproduced (to an extent) the code from [38]. I'll refer to this by it's citation number and as "the original paper" later in the text.
- Taken output from the Momentum Contrast for Unsupervised Visual Representation Learning (MOCO) model into a 2D UMAP plot and quantified the quality of the explanations using 6 different metrics.

Section 8.1 discusses the work and idea of [38], including used technologies. Section 9 discusses my work and how I've configured my pipeline. Section 9.5 shows my output from my model and my explanation metrics. Finally, I discuss these results in Section 9.6 and future work in Section 9.7.

8.1. Background

8.1.1 Summary of the original paper

Popular SSL algorithms such as MOCO (Section 8.2) and SimCLR [6] have a loss function named Information Noise-Contrastive Estimation (InfoNCE). This is an instance discrimination function that uses combinations of positive pairs, an image x and its augmentation x^+, and minimizes the difference between these while maximizing the difference between the image and other tiles x^-. However, the images x^- are sampled from the same batch that the image x is in. The problem that can arise is that there may not be formed negative pairs between tiles from the same slide. Thus the model learns to discriminate on features that may be slide-specific rather than feature-specific for the tumour. This is referred to by [38] as *slide-level batch effects*.

[38] notes, citing [25] that you can tweak the loss function by choosing samples using a given condition. However, this can be difficult to do without introducing bias or *feature suppression*, that optimizing for one feature comes at the cost of another.

Each WSI is large, up to several gigapixels in size. Because of this, each image is usually cropped up into tiles. Unfortunetaly, annotations for each tile is not common, so instead you rely on slide-level annotations. Usually, tile-level predictions are thus pooled together for each slide after processing to get a form of aggregate prediction which is used later. [5] shows an alternative in their DeepCluster algorithm. They first use *k-means* to assigns clusters for each data feature they extract from unsupervised learning methods. Each cluster represents a feature. They then use these features to update their models (convnet).

Results from the SSL are calculated based on a survival-analysis model. The vector v_j (sum of probabilities of tiles from cluster from slide?) is compared with the slide-level ground truth y_j of whether or not the tumour seen in the slide caused recurrence. If there is recurrence, they also use the number of days from observation until a follow-up oberservation t_j. If there was not recurrence, then length of followup-time is used?

In training they use public TCGA and CPTAC datasets of H&E-stained biopsy slides. They observe from previous work that other training algorithms cluster together tiles from the same slides, creating a batch effect where the algo-

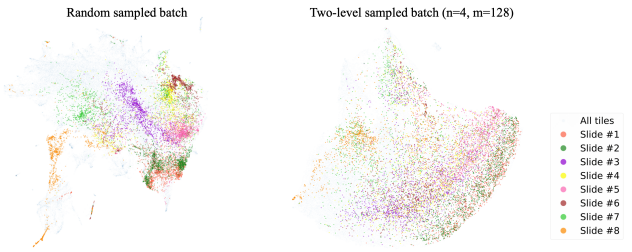


Figure 3. Figure taken entirely from [38]. Both pictures show a 2D UMAP projection of tile representations. On the left a training algorithm with randomly sampled tiles from slides, showing that features learned in the same slide cluster together. Using a different sampling algorithm, the picture on the right shows a more even distribution w.r.t slides, according to the authors

rithm learns features that are more slide specific than feature specific. This indicates that the network might do poorly on generalized input. They visualize this using 2D UMAP, shown in Fig. 3.

The paper finds that the the model with 4 different slides per loss function evaluation in batches of 128 gives a high *C-index*[1] of 0.646 ± 0.055. For experimental evaluation, they compare their own model with a batch sampler that selects opposing tiles x^- from the same slide. They find that their own solution performs slightly better: the same-slide sampler achieves C-index of 0.631 ± 0.052. Random sampling gets a C-index of 0.0601 ± 0.069. Their *brier* scores[2] for the best model are 0.200 ± 0.051 for 2-year prediction.

8.2. SSL model: MoCo

MOCO [15] is a contrastive learner (also discussed in Section 7.1. It uses two underlying models, one *key encoder* which converts an input image to latent space and a *query encoder* which translates negative or opposite pairings to latent space. A core feature of MOCO is that it manages to keep a long *dictionary*, which in layman's terms means it can have many comparison images for its loss functions. In the comparison paper [38], InceptionV4 [32] is used for both encoders. This model has shown good performance on ImageNet, but other papers on LSCC have also used it, e.g. [14].

8.2.1 UMAP

UMAP is a dimensionality reduction algorithm that can be used to visualize high-dimensional data [26]. UMAP works by first making a graph network of all points in its input and then learns an embedding by different techniques. I'll omit the technical details, but what is relevant is that it has elements of randomness for approximation steps. You can produce deterministic results by feeding a *state variable*.

[1] concordance index: ratio of predicted risk and actual recurrence time, higher is better

[2] mean square error from actual recurrence and predicted probability: lower is better

9. Methodology

9.1. Pipeline

I follow the same pipeline as the original paper. First, images are pre-processed from slides into tiles. They are also color normalized using a Vahadane method [34] with a reference image of a tumour WSI stain (image can be found in their online repository). Then, a MOCO model is trained using InceptionV4 models as encoders. A pretrained fully supervised InceptionV4 model is then used to detect which tiles have cancer and which tiles does not. The *query encoder* is then extracted from MOCO and used to embed features from the selected cancerous tiles. These features are visualized using UMAP in Figure 5. The embeddings from the InceptionV4 model are then clustered into 50 different groups (probabilistcly) using GMM. From these groups, a link between survival rate and recurrence is predited using a Cox hazard model [11]. For the survival model, slide-level annotations are also used as a ground truth.

9.2. Recreating Work

The first thing I did when re-creating the original paper was to get the code from their listed website . The code repository has listed commands for re-creating their work. However, I ran into several following difficulties outlined in this section.

9.2.1 Datasets

The original paper's GitHub page lists their sources for finding the images, but not the annotations. For TCGA-LSCC I tried both the official website [2] and the publicly listed "TCGA-Clinical Data Resource (CDR) Outcome" from [3]. Neither dataset was compatible with the code as is, mismatching both on filename and certain columns. By combining different files from the former resource, I was, able to get the right data columns. However, of the more than 500 patients that were in the image dataset, only 170 of those had a defined value in column *new_tumor_event_dx_indicator*, necessary for recurrence prediction.

The code listed by [38] only chooses these images for training, even though the training is unsupervised. So, either their annotation dataset was different than mine, I've misinterpreted something or this is a bug pruned away their input data. Nevertheless, due to hardware capacity issues discussed in Section 9.3, I chose to only use the 170 WSI's,

leaving the code as-is. The CPTAC dataset had similar issues, so I chose to leave it and defer it for future work.

The authors offer a pretrained network. This network is an InceptionV4 model that can be used an encoder in MOCO. However, they do not specify explicitly what the model is trained on nor what hyperparameters they used - assumably it would be the data used in their paper. It does however fit the description in their Appendix A.2: "To only analyze the cancerous parts of WSIs, all the tiles in the tumor slides without any tumor cells were excluded based on a separate Inception-v4 trained with full supervision to distinguish tumor and normal cells (AUC at 98%)". For the model they cite [10]. That paper, however, claims to use Inception-v3, which is without the *residual connections* of its successor. Nevertheless, the pretrained model was compatible with an InceptionV4 structure.

9.2.2 Code

Much of the code did not work out the box. The kind of errors I ran into were missing imports, file and directory names that were mangled (trying to access files in a directory and subsequently looking for the same files in a different directory) and other errors such as API mismatch when trying to load *pickled*[3] files directly into Pytorch. Missing imports might stem from having written code in an environment with hidden files, the file mangling might be because they had duplicate files scattered in their directories and the API errors might arise from using different versions of third-party tools. Be that as it may, it made the process of recreating their work difficult.

There is no code to reproduce their plots and graphs. There is also no metioning in the code or the paper of what hyperparameters and slides that were used for their UMAP plots. This would have been useful to do a verification of the original paper's main claim that they reduce slide batch effects.

9.3. SSL model

I follow suite of the original paper (and [15]) and use 128 embedding dimensions, number of keys from negative pairs K as 65536, temperature as 0.07 and encoder momentum as 0.999. For batch size, however, I ran into several issues. Setting batch size = 1 occupied around 27 GB of main memory on my computer, which only has around 32 GB total memory. When writing the model to disk, it occupied a few additional GB - grinding my computer down to a halt. I tried another machine with 128 GB available, but then the GPU ran out of memory (max capacity 24 GB). This led me to reduce the floating point precision from 32bit to 16bit during training. I still did not have capacity to increase the batch size, however.

[3]serialized Python objects, written to file for later use, see https://docs.python.org/3/library/pickle.html

9.4. Quantifyable Metrics

Testing all different dimensions on UMAP examples is arguably difficult. I tried every metric listed in SciPy's "spatial distance module" [36] and several other inspired from section 2.9 in [9] (which was partially covered in Section 5). The ones I ended up finding the most *indicative* when evaluating different samples were:

1. Euclidean distance: average of metric distances between points
2. Shannon index [27]: a *diversity index* for all points
3. Cosine similarity: angles between points, averaged
4. Hausdorff: given a point in set S, what is the worst distance if chosen by an aversary, that you have to travel to get to a point in another set T?
5. Wasserstein [35]: Also known as the "earth mover distance" because it can be seen as how much earth you would have to move to get one set of points S onto T. It's considered a *distributional distance*
6. Frechet (as per [13]): a similarity metric of *curves*.

The euclidean distance has an intutive interpretation: if the average is small, then all points are probably clustered together. A single point out of bounds can however skewer the number. The same applies to the Hausdorff and Frechet metric, which depends only on supremum or infima. Hausdorff is nevertheless a common set distance metric, and Frechet can deal with some of Hausdorff's weak points, so I've chose to include them nevertheless. Cosine similarity is not as common, but I chose to include it because of it's "simple-to-comprehendedness", similar to euclidean distances. The remaining metrics, Wasserstein and Shannon indices are distribution metrics. These are mentioned in [9] to find representative subgroups of populations, but also have potential utility in this paper.

Frechet and Wasserstein are metrics that depend on having the same number of points in comparison. To compare sets of different lengths, I have used random sampling to choose points from the larger set. I also compute averages between one and one sets, not from one set to all others.

9.4.1 Hyperparameters

Figure 4 shows the loss output from training a model using similar parameters as [38]. It seems to stabilize after the 50 last epochs, although it is possible that an extra few hundred epochs would reduce the loss down even further. This was

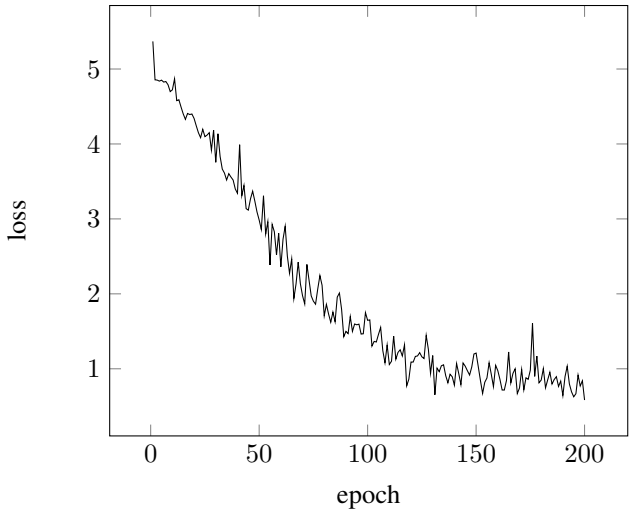


Figure 4. Loss from training model

not done in an effort to keep models as close to the original paper as possible.

For our work, we use *random_state = 42* in the python UMAP API, version 0.5.

9.5. Results

For comparative evaluation, five different models have been trained. Model α is the same as the optimal model from [38], except that the batch size is set to 1 rather than 4 as discussed in Section 9.3. β is a model trained on only 3 photos, γ is a model similar to α except it has only used crop and color distortions as image augmentations (similar as in the recommended section from [6]), δ is a model trained using unconditional SSL, i.e. random sampling of slides and ϵ is a model similar to α, except it's only trained for one epoch. ϵ is only used to a biased, clustered photos for Figure 5.

It can be difficult to tell the different models apart from the figures in Figure 5. ϵ is the only one to undubitously stand out with clear clustering from each slide. The numbers from Table 1 capture this in the Hausdorff, Frechet and Wasserstein distance that are all significanly higher for ϵ than for any other trained model. The cosine similarity stands out in model γ. This *may* be used as an interpretation that the dataset is less biased, however, the results section in Table 2 are still low for γ compared to others.

From Table 2 we see that β is the model with the highest C-index. This is highly unexpected as it only has had three images for it's input during training. It's brier-score, is worse than for all others. This could indicate that a C-index is not sufficient alone for model evaluation, unless the three slides for training were somehow entirely representative.

The five-year brier score are significantly higher than the two-year for all four models. The biggest discrepancy is for β that jumps from 0.23 to 0.39. The most *stable* model in this sense is model α that only jumps from 0.16 to 0.23. While 5-year brier scores was in the code from the original paper, it was not reported in the paper.

9.6. Discussion

To make the results more reliable, multiple test and validation sets could be used to provide a confidence interval. It has, however, been omitted in this paper as my overall goal with this was *not* to get a high score for LSCC training but rather an evaluation of the XAI UMAP explanations.

Models α, β, γ and δ did not in my opinion provide UMAP figures that were signifcantly different from each other. This came as a surprise as the parameters from different models were quite different in both augmentations and input size. For a more thorough analysis, especially with consideration of Quantus [16], I should have used more obscure hyperparameters.

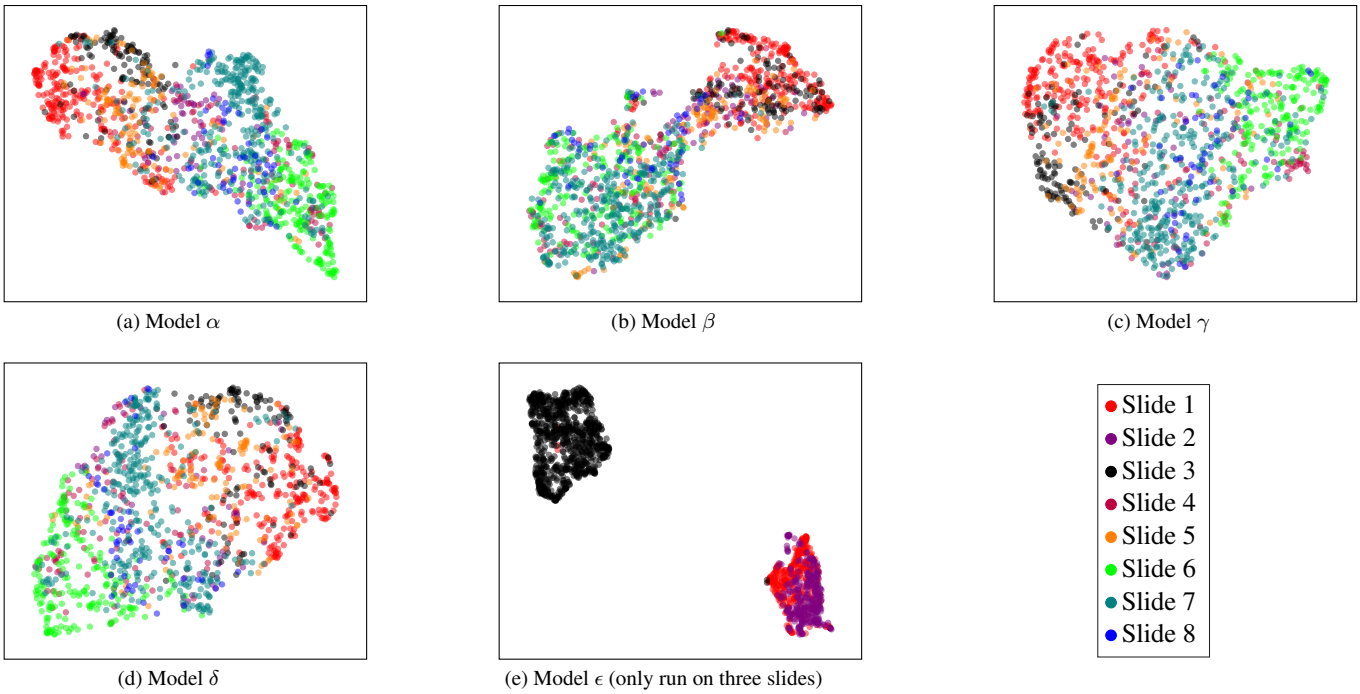
I would expect β to be more clustered together since it was in a sense overfitted to its limited inputs, but this does not seem to be the case by both C-index and distance scores. It could be that it's dictionary of negative pairs was large enough that it learnt generalizable features.

The different metrics seem to provide *some* confidence for UMAP, but not signifcantly. The Hausdorff, Frechet and Wasserstein distances can all indicate whether or not a model's output is heavily clustered and distanced from other sets, as seen for model ϵ. To really assess the numbers, there should have been more model experimental evaluations from my end - but I was unable to find these in time.

While the slides were randomly chosen, it would be interesting to compare them for images that had different classes such as one image being tumor-free vs another that had tumours. This should ideally show some clusters that were far apart in the UMAP figures.

9.7. Future work

There are many ways to compare different different model outputs. One is to calculate numbers for more (or all) combinations of slides. This is also feasible since the time to compute metrics and UMAP models is only a few seconds on regular desktop computer. However, the numbers are there to interpret the *visual* representation. Thus, in order to confirm whether or not the averages are useful, they should be experimentally validated on multple datasets, which could detect overfitting. Another consideration is how the points are sampled. Right now I have used averages between individual sets (slides) for Hausdorff, Frechet and Wasserstein. Another sampling method could be to compare these metrics for randomly sampled point from all other sets/slides.



(a) Model α (b) Model β (c) Model γ

(d) Model δ (e) Model ϵ (only run on three slides)

Figure 5. UMAP plots from 8 randomly selected tiles from the validation set used in model training. The samples are the same for all graphs

model type	α	β	γ	δ	ϵ
Hausdorff	3.84	3.06	3.26	3.57	12.98
Frechet	5.39	5.7	4.67	4.88	17.84
Wasserstein	3.37	3.17	2.77	2.75	11.74
Euclidean	2.15	2.51	2.2	2.36	2.12
Shannon	0.06	0.07	-	0.07	-
Cosine	0.025	0.02	0.45	0.02	0.029

Table 1. Quantitative measures/means from the UMAP figures in Figure 5. *Shannon* would not be computed for some sets, thus left blank

model type	α	β	γ	δ
C-Index	0.53	0.63	0.35	0.32
Brier (2 yrs)	0.16	0.23	0.21	0.16
Brier (5 yrs)	0.23	0.39	0.34	0.25

Table 2. Brier and C-index scores from the validation set

9.8. Conclusion

In this paper, we have experimented and evaluated model from [38]. Some of the work was difficult to reproduce both from unexpected errors in the code and harware limitations. However, five models were trained and compared, both for results and for a UMAP evaluation, which was both quantitatively but also visually interpreted. Overall, it seems that some metrics like Hausdorff, Frechet and Wasserstein *might* be good indicators of overfit, but more work is needed to evaluate it thoroughly.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Explanations can be manipulated and geometry is to blame. *arXiv preprint arXiv:1811.10158*, 2018. 3

[2] The Cancer Genome Atlas. The cancer genome atlas lusc clinical data. https://portal.gdc.cancer.gov/legacy-archive/search/f?filters=%7B%22op%22:%22and%22,%22content%22:%5B%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22cases.project.program.name%22,%22value%22:%5B%22TCGA%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22files.data_category%22,%22value%22:%5B%22Clinical%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:

%7B%22field%22:%22cases.project.primary_site%22,%22value%22:%5B%22Lung%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22cases.project.project_id%22,%22value%22:%5B%22TCGA-LUSC%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22cases.project.disease_type%22,%22value%22:%5B%22Lung%20Squamous%20Cell%20Carcinoma%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22files.data_format%22,%22value%22:%5B%22Biotab%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22files.data_type%22,%22value%22:%5B%22Clinical%20data%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22files.access%22,%22value%22:%5B%22open%22%5D%7D%7D,%7B%22op%22:%22in%22,%22content%22:%7B%22field%22:%22files.platform%22,%22value%22:%5B%22Clinical%22%5D%7D%7D%5D%7D, Accessed 2023-02-17. 7

[3] The Cancer Genome Atlas. The cancer genome atlas pancan atlas. https://gdc.cancer.gov/about-data/publications/pancanatlas, Accessed 2023-02-17. 7

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 2

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2018. 6

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 4, 6, 8

[7] Prakash Chandra Chhipa, Richa Upadhyay, Gustav Grund Pihlgren, Rajkumar Saini, Seiichi Uchida, and Marcus Liwicki. Magnification prior: A self-supervised method for learning representations on breast cancer histopathological images, 2022. 4

[8] Yu-An Chung and James Glass. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP*, 2020. 2

[9] Line H. Clemmensen and Rune D. Kjærsgaard. Data representativity for machine learning and ai systems, 2022. 4, 8

[10] Nicolas Coudray, Paolo Santiago Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Fenyö, Andre L Moreira, Narges Razavian, and Aristotelis Tsirigos. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature medicine*, 24(10):1559–1567, 2018. 7

[11] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972. 7

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 3, 4

[13] Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. 1994. 8

[14] Martin Halicek, Maysam Shahedi, James V Little, Amy Y Chen, Larry L Myers, Baran D Sumer, and Baowei Fei. Detection of squamous cell carcinoma in digitized histological images from the head and neck using convolutional neural networks. In *Medical Imaging 2019: Digital Pathology*, volume 10956, pages 112–120. SPIE, 2019. 6

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. 6, 7

[16] Anna Hedström, Leander Weber, Daniel Krakowczyk, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina Marina M.-C. Höhne. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond. *Journal of Machine Learning Research*, 24(34):1–11, 2023. 3, 9

[17] Vetle Hofsøy-Woie. Distributing deep learning on hpc. 4

[18] William Kruskal and Frederick Mosteller. Representative sampling, i: Non-scientific literature. *International Statistical Review / Revue Internationale de Statistique*, 47(1):13–24, 1979. 3

[19] William Kruskal and Frederick Mosteller. Representative sampling, ii: Scientific literature, excluding statistics. *International Statistical Review / Revue Internationale de Statistique*, 47(2):111–127, 1979. 3

[20] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. 5

[21] NRIS. Hpc intro: When one server is not enough. https://training.pages.sigma2.no/tutorials/gpus-on-hpc/episodes/11-hpc-intro.html, Accessed 2023-02-17.

[22] OpenAI. Openai. https://openai.com/, Accessed 2023-02-17. 2

[23] Yuechao Pan, Muhammad Osama, and John D Owens. Synchronous vs. asynchronous gpu graph frameworks. In *The 7th Workshop on Multi-core and Rack-scale Systems*, 2017. 4

[24] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre,

Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022. 1

[25] Joshua Robinson, Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Can contrastive learning avoid shortcut solutions?, 2021. 5, 6

[26] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021. 6

[27] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 8

[28] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020. 2

[29] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. 2

[30] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, Oct. 2017. 1

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*, 2014. 2

[32] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016. 6

[33] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, 2011. 3

[34] Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE transactions on medical imaging*, 35(8):1962–1971, 2016. 7

[35] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969. 8

[36] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 8

[37] Xuhui Zheng, Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Discovering faster matrix multiplication algorithms with reinforcement learning. *Advances in Neural Information Processing Systems*, 32:2234–2245, 2019. 1

[38] Weicheng Zhu, Carlos Fernandez-Granda, and Narges Razavian. Interpretable prediction of lung squamous cell carcinoma recurrence with self-supervised learning, 2022. 5, 6, 7, 8, 10