A Comparative Study of Pre-Trained ConvNeXt and Vision Transformer Models for Image Classification

Stuart Groves
School of Computer Science & Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
2356823@students.wits.ac.za

Hairong Bau
School of Computer Science & Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
hairong.bau@wits.ac.za

***Abstract*—This paper presents a comparative analysis of pre-trained ConvNeXt, Vision Transformers (ViT), and Hierarchical Vision Transformers with Sliding Windows (SwinT) for image classification. Drawing inspiration from successful strategies in SwinT, ConvNeXt emerges as a dominant successor to traditional convolutional neural networks (CNNs). Evaluations on diverse datasets, including satellite imagery, medical X-rays, and a dataset discerning real and generated images, consistently demonstrate ConvNeXt's superior performance. Remarkably, ConvNeXt's enhanced capabilities are achieved without significant variations in training times and memory requirements. This study positions ConvNeXt as an optimal backbone for image classification, affirming its potential across applications such as satellite analysis, medical screenings, and media authenticity detection.**

I. Introduction

Vision Transformers (ViT) models have risen to prominence within computer vision in recent years [11]. Their original application was an entirely unrelated field, natural language processing (NLP), but despite that, they swept traditional convolutional neural networks (CNNs) throughout subdomains of computer vision. Subsequently the ViT, which initially changed very little from its NLP counterpart other than the introduction of a 'patchify' layer [22], was improved upon in the creation of a architecture formally known as a 'Hierarchical Vision Transformer Using Shifted Windows' (SwinT) [21]. The purpose of which was to address the scalability, or lack thereof, of the original ViT. The researchers largely succeeded and proved their model through the accuracy on the ImageNet-1k benchmark [16]. However transformers had yet to face their final test against the convolutional neural network. Following SwinT and ViT, a group of researchers aimed to prove that despite their new found dominance, more could be learnt from the transformers' dominance that could be reapplied into a newer generation of convolutional neural networks (CNNs) to reclaim their dominance. This newer generation CNN, dubbed ConvNeXt [22], did just that. In a dominant showing, the researchers implemented successful strategies learnt from the SwinT in order to modernize the CNN as a whole, performing better than the SwinT across multiple benchmarks of computer vision [22].

The aim of this paper is to provide further proof of the dominance ConvNeXt has against the transformer architectures, ViT and SwinT. In order to do this, we will be evaluating each architecture's capability to perform as the general backbone of image classification specifically. We will test how well pre-trained models of these architectures can adapt to new domains by retraining them across unseen datasets in evaluation of their feature extracting capabilities.

These three datasets were not chosen at random, but rather they are datasets we believe to be of importance in a modern context. Firstly, the UC Merced Land Use dataset [34]. This dataset is meant to be an illustration of the power of a modern architecture's ability to classify remote satellite imagery in a modern era of aerial surveillance applications ranging from surveying to military applications. Secondly, a chest x-ray dataset [18]. This dataset is meant to simulate a modern architecture's ability to be used as potential diagnoses screening on large scales that surpass what is capable by individuals at a macroscopic level. While it's unknown whether neural networks will ever be allowed/able to diagnose individuals without medical professional supervision, when used as a screening tool, it may save countless lives of unsuspecting individuals whom are symptom-less at the time or if there is a shortage of medical professionals capable of giving a diagnosis in a given area. Lastly, the CIFAKE dataset [4]. This dataset uses a combination of images obtained from CIFAR-10 [7], and images generated by stable diffusion which are intended to be as close to real as possible [28]. In an age where misinformation appears to be a worsening crisis, it's becoming popular to believe none of what you see, and none of what you hear from unverified sources. The problem is being worsened by AI generated media ranging from photographs to audio recordings. It may soon, as AI progresses, become increasingly likely that a majority of people will inevitably be consistently fooled by said generated media which may in turn lead to misinformation by bad actors on anonymous forums. Creating AI which are capable of dissecting generated images, audio and written text may prove crucial to filter these harmful generated medias before they spread misinformation. This is what our third dataset is to simulate; how effective modern architectures are at differentiating generated images from real ones.

Initially, we evaluate the average case of each architecture by using a subset of hyper-parameters which are semi-optimal across all of the architectures in order to gauge how well they perform in cases other than those with optimal fine-tunings. Across all datasets the models created and tested, are testament to the power of ConvNeXt as a successor to the traditional CNNs.

Following those initial models, we took the best model of each architecture for each dataset and trained on a much higher epoch count to ensure convergence. Once again, across all metrics, ConvNeXt outperformed both transformer-based architectures by large margins.

Thus we conclude that ConvNeXt is better suited as a backbone for, at the very least, image classification tasks than either transformer-based architecture based on its dominant performance. Similarly there is little to no difference across training times and memory usage of each of these models, further illustrating the architecture's dominance as a general backbone feature extractor.

II. Related Work

The study was mainly inspired by the ConvNeXt introduced in 'A ConvNet for the 2020s' [22]. In addition, the study considers two transformer-based architectures, namely, the Swin Transformer (SwinT) [21] and its predecessor the Vision Transformer (ViT) [11]. The purpose of ConvNeXt was to illustrate that although, since the introduction of the Vision Transformer in 2021, transformer-based models had been dominant in the field of Computer Vision, that there was hope yet for convolutional neural networks (CNNs). The researchers of ConvNeXt [22] pointed out that the design of fully connected convolutional neural networks had some inherent advantages in the field. They proved the superiority of ConvNeXt, in scalability and translational-equivariance, as the proposed architecture fully outperformed the SwinT. This finding shows that CNN-based architectures may still be a better backbone for Computer Vision than the transformer-based models [22]. Following this, and observing similar research in works by Shenglong Chen, Yoshiki Ogawa, and Yoshihide Sekimoto [5], as well as Manav Garg et al. [13], we aim at employing similar experimental methods in order to illustrate that ConvNeXt has superior performance in image classification than both transformer-based models.

III. Method

This section serves to elaborate on dataset selection, and design choices made for the execution of this research as well as an overview of the components on which the research is founded.

A. Dataset Selection

This paper uses three distinct datasets, which each cover a particular domain within image classification. The specifics of each dataset is described below. The choice of these datasets was to cover as much real world application as possible to prove the application of the architectures in question. While we know they are powerful architectures given their high accuracy on ImageNet 1k/21k [16], the specific application of these networks is a judgement of how well they are able to transfer knowledge from their pre-trained domain and the effectiveness of their feature extraction across multiple unseen domains.

1) Data Augmentation: Data augmentation is specified for each of the datasets, with the first dataset being unique in augmentation to the remaining two. It is a deliberate design choice not to augment the data further than what is inherently present within the first dataset in order to gauge the maximum performance on relatively un-augmented, easily classified data. Hence this makes the latter two technically harder due to various augmentation techniques used. The latter two datasets used throughout this report, share the same myriad of data augmentation techniques. Data augmentation was not applied to the 'test' and 'validation' segments of each of these datasets and only the 'train' segment.

2) Default Transform: The default transform is one that all three datasets utilize and simply resizes every input image to 256×256 using bilinear interpolation [3]. Subsequently it then crops to the center of the image of size 224×224 and converts the image to a tensor. Lastly the tensor is normalized using the ImageNet [27] MEAN and STD which in turn ensures the images are in similar visual intensity ranges to those on which each of the networks were pre-trained. This normalization is crucial, as without it, each network would have difficulty in extracting the features consistently on which they were pre-trained.

3) Training Transform: The latter two datasets use a training transform that differs substantially from the default transform. Initially the image is randomly flipped both horizontally and/or vertically with $p = 0.5$ for each. Again, similar to the default transform, the image is then resized, but then instead randomly cropped to a dimension of 224×224. Similarly it is then converted to a tensor and normalized using the ImageNet [27] MEAN and STD. Lastly, noise is added to the image with a chance, $p = 0.1$, that any given value in the tensor is set to a random value in the range $[0, 255]$. This noise will challenge the architectures. In the case of vision transformers, which struggle with translational-equivariance [22], but also the ConvNeXt as we've seen possible one pixel attacks result in extreme mis-classification in neural networks [31]. However those one-pixel attacks were adversarial, so it is unlikely but interesting nonetheless.

4) Dataset 1: UC Merced Land Use: Satellite or bird eye's imagery is an important subset of the image classifiction domain. The applications range anywhere from surveying statistics to military applications. The dataset is a well benchmarked dataset known as UC Merced Land Use Dataset from a paper regarding land use classification [34, 36]. The choice of this dataset was motivated not only by its importance contextually, but also as a multi-class domain that would provide a deeper understanding of how well each of these models scale in confidence to classifying multiple classes. This is observed in their relative loss metrics later.

For this dataset in particular, a specific augmented version was found on Kaggle, produced by Gotam Dahiya [20], which provided some basic data augmentation such as rotating in order to artificially increase the amount of images per class from 100 to 500. Each image was augmented exactly four times and originals preserved in order to not create an imbalance of specific images within a given class.

This is the 'easiest' dataset of the three, despite being a multi-class classification problem. The intent is to gauge solely how a model scaled with classes rather than with both classes and augmentation, hence only light augmentation is used to artificially increase the amount training data present.

5) Dataset 2: Chest X-Ray Classifcation: Medical Imagery Classification is one of the most promising Image Classification domains. However it is also the most scrutinized, and for good reason. There is little to no room for error in medicine; an incorrect diagnosis could very easily cost lives so therefore accuracy in this field is imperative. It is unknown whether or not Image Classification will ever become approved to be used in the medical field but it is possible that it may be used for screening large groups of people where individual attention from a clinician may be impractical or impossible. Nevertheless it is yet to be seen whether it will solely make diagnoses without the corroboration of a qualified professional.

The dataset of choice is a one made publicly available from a paper that examined diagnoses by image-based deep learning [18]. While the original dataset sought other classification classes to the ones we've used, we settled on a dataset that is a binary classification subset of the original which classifies between Pneumonia and Normal chest-xrays. Found on HuggingFace [17], the datset is unique in its class imbalance which is useful in discriminating a model which may struggle with class imbalances. While our loss function inherently combats this via label smoothing (elaborated later), it is noteworthy nonetheless and may prove useful in evaluations. The dataset is split 73 and 27 between the Pneumonia and Normal classes respectively.

This dataset is anticipated to be the middle-ground of the datasets in terms of classification difficulty. Additionally it introduces data augmentation which will lower training accuracy but should hypothetically improve the models confidence on test data.

6) Dataset 3: CIFAKE: Image Generation is becoming more accurate by the day with new advances in algorithms such as Stable Diffusion making incredibly deceiving images. Two recent examples of these images being used to feign images of popular figures (Former US President Donald Trump being arrested and the Pope breaking the internet with 'his' fashion choices [14, 12]) are proof of this. It wont be long before AI generated images could be maliciously used to spread misinformation on public platforms by anonymous users and bad actors. We believe its appropriate then that the third and final dataset is a binary classification task named CIFAKE [4, 28], which is a large dataset that uses real-world CIFAR-10 images [8] and images generated by Stable Diffusion to create the REAL and FAKE classes respectively.

The dataset was specifically obtained from Kaggle [6], and was uploaded by the author so no special alterations have been made from the dataset used in their paper. This dataset, given the nature of the domain not being so clear cut and with Stable Diffusion being deliberate in generating realistic images, is anticipated to be the hardest of the datasets to classify. Additionally we also use training augmentation on this dataset which should lower the training accuracy but again improve the confidence of the model on test data.

In light of the limitations [IV,E] of this report, this dataset will be shrunk significantly from its original size due to processing time constraints. Despite this, it is still the largest dataset of the three and has balanced classes.

B. Architectures

It is no secret that within the past decade, Image Classification models have improved exponentially [7]. With a clear example being the rate at which models have improved on CIFAR-10, initially, breakthrough models were scoring an error rate as high as 21.1% [19]. Today we have models such as one of the models chosen for our comparative analysis, ViT, scoring as low as 0.5% [11]. This is roughly halving the error rate once every two years. It's more common to see breakthrough papers today being published with what were once considered marginal gains, seeking to improve by just fractions of a percentage. Thus it is important that we strictly evaluate these models and ensure that these marginal gains have a strong basis and do not sacrifice efficiency and/or scalability in order to maximize a few fractional percentage points.

The architectures we have selected to evaluate on our outlined dataset are breakthrough modern architectures at the forefront of image classification as well as other image related domains, with transformers additionally being dominant in the field of Natural Language Processing. However with marginal gains comes the question whether or not they hold up across unseen domains. Using transfer learning methods and pre-trained base models for each of these architectures, we will evaluate whether or not they can apply knowledge to novel domains and hence classify to the same degree they can in their pre-training domain of ImageNet [16].

1) ViT: Vision transformers, despite originally being from an entirely different field of Natural Language Processing, recently have proven themselves dominant over traditional CNNs [11]. Despite the seemingly apparent lack of changes to the architecture, besides a 'patchify' layer [22], which segments an image into smaller patches over the image, a slightly modified architecture from an entirely other field performed incredibly well at image classification.

The authors of the first vision transformer implementation [11], proved that while previously transformers had little to no part in computer vision domains, other than occasionally being used to supplement CNNs, that a newly proposed ViT could perform as well if not better than traditional neural networks while also being more efficient to train. This paper had a knock on effect and subsequently the two other architectures following the ViT were direct results of this advancement.

The architecture is as follows [11]:

- Patchify layer: The ViTs first layer uses segmentation to split an input image into a grid of 'patches' usually 16×16 pixels large. In the context of transformers in Natural Language Processing, this serves as a 'bag of words' for the network where each 'word' is instead any given patch. This is the main difference between the original transformer architecture and the now ViT.
- Patch vectorizing: Each patch is subsequently transformed into a flattened vector which then 'contains' the features of any given patch.
- Transformer encoder: The transformer encoder, a stack of self-attention layers [35, 2], allows the model to learn long-term relationships between features of the patches.
- Transformer output: The transformer then results in a sequence of vectors of extracted features from the image where finally,
- MLP head: the output of the transformer is given to a Multi-Layer Perceptron which then provides a prediction.

Additionally given by the model overview diagram, Fig. 1, from the authors [11]:

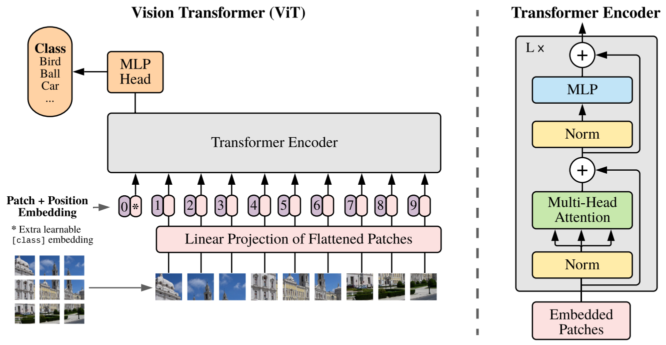

Fig. 1. ViT Model Overview

2) SwinT: Similarly to the ViT, SwinT (formally known as a Hierarchical Vision Transformer using **S**hifted **Win**dows [21]) also has major architectural elements used from transformers. However it was based on the previous work of the ViT [11] and then improved with the purpose of providing scalability to high scale resolutions of images which is something that the ViT did not scale well with.

Their proposal seeks to lower the computational scalability requirements of typical transformer architectures, which do not translate well to computer vision tasks. Unlike text, where each token may depend on the entire sequence, images can be processed more efficiently by focusing on local regions. Therefore, the SwinT uses 'shifted windows' to compute self-attention locally, which reduces the computation complexity and allows for hierarchical feature representation. This also enables the SwinT to create larger and deeper models that can handle higher-resolution images and achieve state-of-the-art results on various vision benchmarks. The architecture is best illustrated by the diagram, Fig. 2, provided by the authors [21], which illustrates the structure of consecutive swin transformer blocks as well as the multi-head self attention modules; W-MSA and SW-MSA for regular and shifted windows respectively.

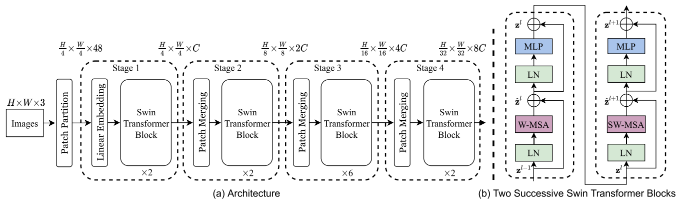

Fig. 2. Swin Model Overview

3) ConvNeXt: Lastly, the model which motivated our research, ConvNeXt [22]. ConvNeXt was a direct result of the SwinT Architecture which seeks to implement properties that were once unique to the SwinT and use them to fully 'modernize' a Convolutional Neural Network to compete with SwinT.

The translational-equivariance of a CNN is something that transformers notably lack [22]. This allows the network to recognize features at various scales and/or positions in an image. Not only is this a more robust model overall in theory but it was also able to illustrate the dominance in practice.

The architecture itself is fully comprised of convolutional networks. In addition to borrowing properties from SwinT, the architecture uses a custom ConvNeXt block. The ratio of these blocks throughout each stage is 1:1:9:1, and overall uses a larger kernal size in comparison to SwinT (from 4×4 to 7×7) for the Patchify layer [5]. Lastly in opposition to a traditional ResNet, an inverted bottleneck structure is used. Another notable difference is also the use of a GELU activation function in place of a traditional ReLU activation function. The ReLU and GELU activation functions are given by the following equations respectively:

$$ReLU(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

$$GELU(x) = xP(X \leq x) = x\phi(x) \forall X \sim \mathcal{N}(0, 1)$$

Lastly the network structure is effectively illustrated in Fig. 3, from a related paper [5]:

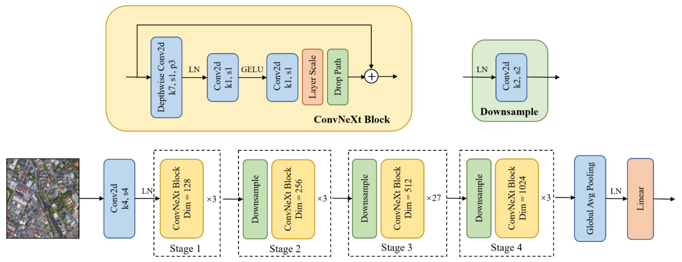

Fig. 3. ConvNeXt Model Overview

C. Optimizer

The Optimizer of choice is AdamW [23], which introduces weight decay aiming to prevent over-fitting by encouraging the weights to remain small which in turn improves generalization. It has also been shown to provide increased stability and converges more quickly than other Optimizers which is helpful given the limitations [IV,E] outlined in this report. The Optimizer remained consistent throughout training as it is a powerful choice in and of itself, and is tried and tested with all the modern architectures in this report.

D. Criterion

The Criterion, namely the Label Smoothing Cross Entropy Loss function from the python package ’timm’ [24], was used to correct class imbalances across the x-ray binary-classification dataset while keeping the loss metric abjectly similar enough that inferences could be made across the novel domains’ loss metrics without class imbalances.

During later evaluation, a base Cross Entropy Loss function without label smoothing was utilized to provide an additional perspective on loss across the architectures for all datasets. The base Cross Entropy provides a clearer metric as to how close the model actually was to the ground truth [15, 25] as labels are no longer smoothed, hence its addition in later evaluation.

E. Scheduler

A learning rate scheduler, in this case a StepLR scheduler from Pytorch [30], was used in conjunction with the AdamW Optimizer and specified learning rate. The scheduler takes three parameters, namely the optimizer, step size and gamma wherein after training for epochs equal to step size, the learning rate is decayed by a factor equal to gamma. This approach reduces a model’s chance of over-fitting by allowing the model to converge more quickly at the start but avoid overshooting optimal parameters toward the end of training with a much lower learning rate.

The settings for the scheduler remained consistent all throughout testing as it was not considered for hyper-parameter tuning after preliminary testing showed that a ’middle of the road’ setting was reliable across all architectures (as optimal epoch settings were found within the range 5-10) with the step size for halving being set to 7, and the coefficient remaining at $\gamma = 0.1$.

F. Statistical Gathering

Many results were gathered through multiple means, however due to the quantity a majority of them have been omitted as they aren’t as convincing as others. The full set of collected data can be found inside the excel spreadsheet found within the GitHub repository provided. Any outliers however have been included as a means of fair testing, we ensured that any surprising results be they positive or negative are analysed.

IV. Results

Herein we will outline our results and discuss any findings. Conclusions will be made later.

A. Comparisons Preamble

To begin testing our models, we first created a subset of hyper-parameters to test all the models across all the datasets. Using the subsets of epochs, learning rate, and batch sizes of $E = [5, 10], LR = [1e-4, 1e-3]$, and $BS = [64, 128]$ respectively, we train 8 models per architecture for each dataset. This initially gives us $8 \times 3 \times 3 = 72$ total models to compare. Obviously that number is exhaustive to some degree so we will initially compare models by their average performances across hyper-parameters which are better and worse for each architecture to see how they perform in general. While technically un-optimized, it still gives us a rough idea of how well an architecture fairs overall when comparing them averaged together by architecture. Following this, we will take the best performing hyper-parameters for each model and carry them forward to 25 epochs in order to ensure convergence and potentially gauge how a model fairs at resisting over-training. Although we already have multiple methods in place such as the Optimizer and the Scheduler to combat over-training, hence it is unlikely we will see any egregious over-training.

Results will continually be analysed by dataset. This structure allows us to compare models fairly as different classification domains cannot be compared directly in good faith.

B. Land Use Dataset

This dataset, as described in the Method, is the only one of our datasets to have multi-class classification. The latter two are binary classification problems. This dataset was also not augmented so we expect it to be the easiest of the datasets with all models achieving their highest accuracy across the board.

Upon testing the subset of 8 models per architecture we obtain the following box and whisker plot which show the range of accuracy achieved by all 8 models of a given architecture across the land use dataset. Immediately, in Fig.

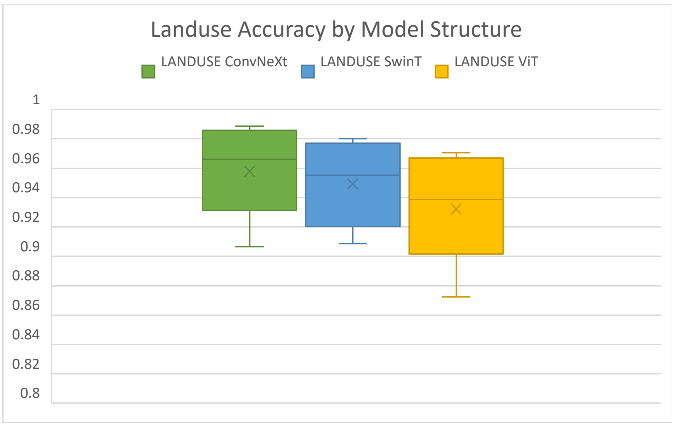

Fig. 4.

4, we see a clear trend between all three models, while very close cut, we observe that as expected ConvNeXt comes out on top with best model, best average, and best median. A little closer is the weakest Swin Transformer model which did perform slightly better than the weakest ConvNeXt model.

Following these observations we take the strongest set of hyper-parameters for all architectures across the dataset and retrain them for 25 epochs each. Coincidentally, throughout the paper, the best hyper-parameters for every architecture were in some cases tied first, but across the board featured a learning rate and batch size of $LR = 1e-3$, and $BS = 64$ respectively. That being said, each of our three best models shared the same hyper-parameters. Following the retraining, during the 25 epochs we observe their relative validation losses. The validation losses, in Fig. 5, provide a clear story

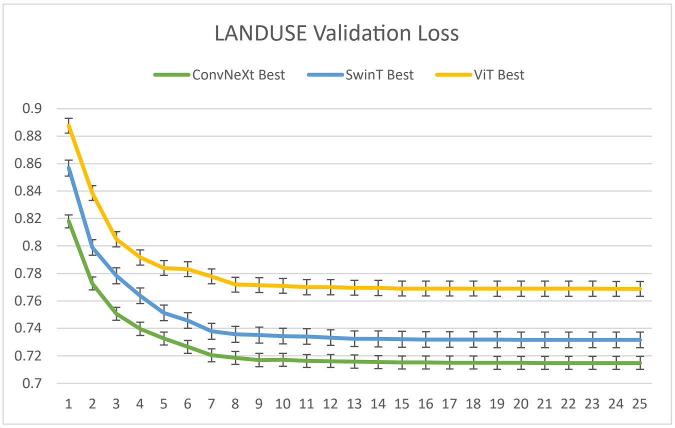

Fig. 5.

into the performance across this dataset. Comparing their validation losses with the loss function being Label Smoothed Cross Entropy Loss, we see that ConvNeXt comes out on top, with the Swin transformer slightly behind and the Vision transformer lagging severely behind by comparison. However the whole story cannot be accurately told by validation loss alone. Hence we observe the relative loss functions on the test data. Thus we now begin judging unseen data performance. Within Fig. 6, a clear trend is observed between the models

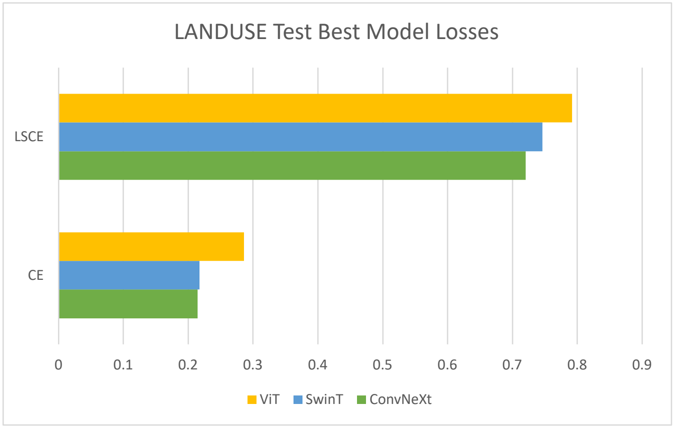

Fig. 6.

again in comparison of the Label Smoothing Cross Entropy Loss, however Cross Entropy without Label Smoothing is a harsher critic of how close a model is to the ground truth. In the case of Cross Entropy we see ConvNeXt and the Swin transformer go neck and neck with a fractional lead by ConvNeXt. Lastly, we observe test accuracy, f1 score, precision and recall: All four metrics within Fig. 7 provide a

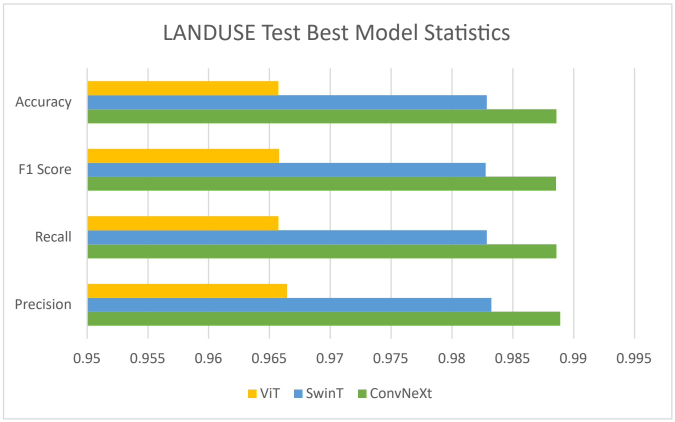

Fig. 7.

better view of just how well each model performed. ConvNeXt is clearly in the lead across the board, with the SwinT close behind and again ViT lagging considerably by comparison.

Across the Land Use dataset we see that ConvNeXt performed the best across training, validation and test subdomains with SwinT coming second and ViT being last.

C. X-Ray Dataset

This dataset, as described in the Method, is the only one of our datasets to have a prominent class imbalance. The other two are perfectly balanced across all classes. This dataset was subjected to augmentation however the domain itself is not 'adversarial' in nature unlike the last dataset CIFAKE [4] which uses Stable Diffusion [28] in order to create difficult to distinguish classes, which are exaggerated by the low image resolution. Given that, we expect this dataset to reveal any problems these models may experience with class imbalances but simultaneously be the 'middle-ground' between the three datasets.

Upon testing the subset of 8 models per architecture we obtain the following box and whisker plot which show the range of accuracy achieved by all 8 models of a given architecture across the x-ray dataset. Immediately upon inspection of Fig 8, we see a clear performance degradation on the SwinT models and ConvNeXt models compared to the ViT models. This is especially true for the the SwinT models which now fall below ViT across best accuracy, average, median, and worst accuracy. ConvNeXt barely retains a lead on stability and all metrics indicative by the extremes of the plot as well as how dense it is. This is important as the transformers are notably unstable compared to the CNN in this context.

Once again we take the strongest set of hyper-parameters for all architectures across the dataset and retrain them for 25

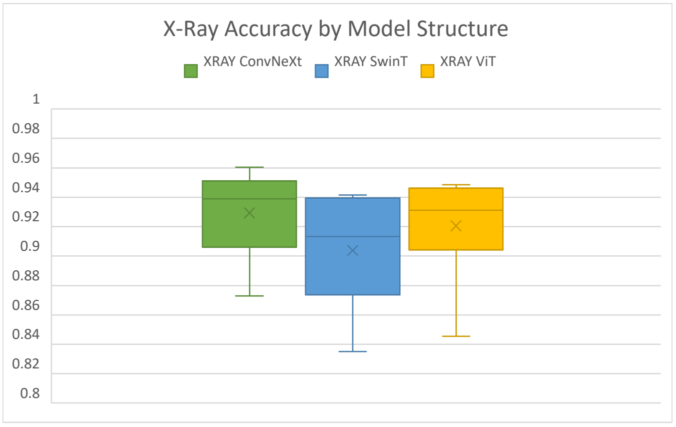

Fig. 8.

epochs each. In this case the strongest are yet again a learning rate and batch size equal to $LR = 1e-3$, and $BS = 64$ respectively. Following the retraining, during the 25 epochs we observe their relative validation losses. The validation

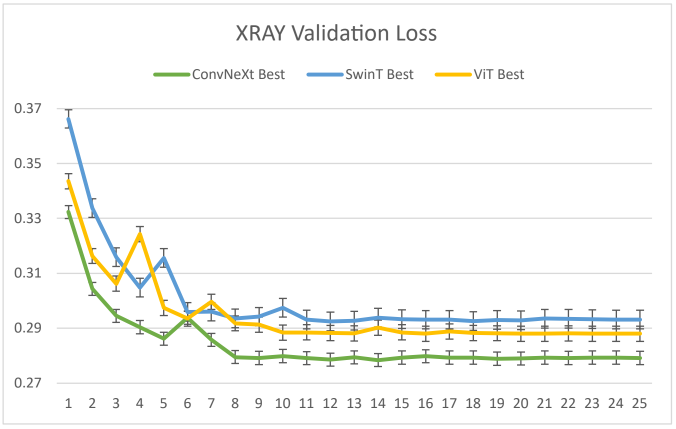

Fig. 9.

losses, given by Fig. 9, provides insight into the stability of models across this dataset. Comparing their validation losses with the loss function being Label Smoothed Cross Entropy Loss, we see that ConvNeXt comes out on top, with ViT coming in second surprisingly and SwinT lagging behind. ConvNeXt appears to be much more stable overall compared to the transformers. We now will observe the relative loss functions on the test data. Thus we now begin judging on unseen data performance. Again, we observe in Fig. 10 that SwinT appears to perform worse than the base ViT. While ConvNeXt maintains its lead it is interesting to note that ViT scored comparatively close to what is supposed to be a dominant model, whereas SwinT, built to improve upon the base ViT has fared more poorly than its predecessor. Lastly, we observe test accuracy, f1 score, precision and recall: Shown by Fig 11, all four metrics show ConvNeXt performs incredibly well. However we observe that the test accuracy of ViT and SwinT are identical in this case. Given their loss metrics,

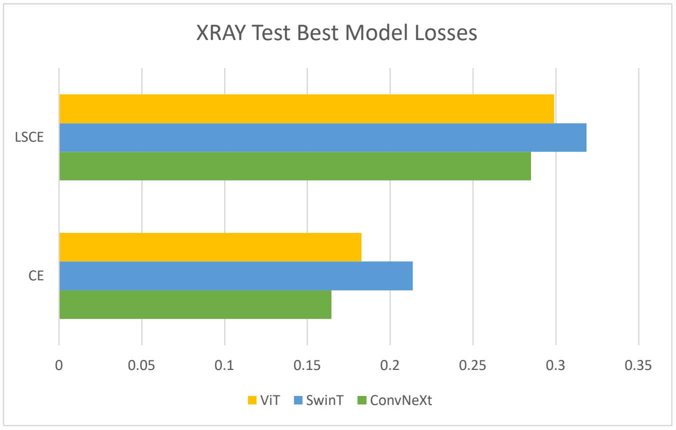

Fig. 10.

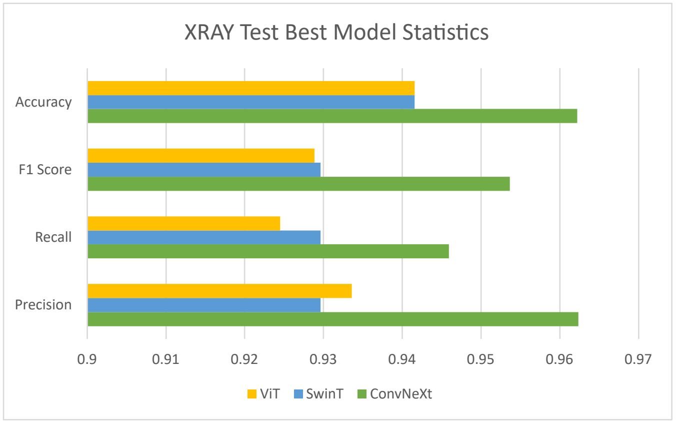

Fig. 11.

SwinT seems to be performing better than expected. We note that there is a clear imbalance in recall and precision for ConvNeXt as well as ViT; this imbalance is not present in SwinT however. This could be a clue to the loss function performing unexpectedly with regards to class imbalances on this architecture specifically.

Across the Land Use dataset we see that ConvNeXt performed the best across training, validation and test subdomains yet again. However an unexpected turn of events has shown that SwinT is not so easily dominant over its predecessor.

D. *CIFAKE Dataset*

This dataset, as described in the Method, is expected to be the most difficult for all these architectures. As previously mentioned, the dataset is comprised of two classes, REAL and FAKE. The challenge of this dataset is the fact that there is no apparent difference between the classes if Stable Diffusion does a good enough job. The classification comes down to artifacts and/or other inconsistencies that may occur with generative images. To complicate this further, the resolution is incredibly low throughout this dataset meaning that looking for small details may prove fruitless. This classification is meant to challenge each architectures' ability to extract features.

Upon testing the subset of 8 models per architecture we obtain the following box and whisker plot which show the range of accuracy achieved by all 8 models of a given architecture across the CIFAKE dataset. Suprisingly we observe in Fig. 12, ConvNeXt has been dethroned in its general performance. Scoring lower on mean, median and best case than both the transformer based models. Additionally its worst case scenario is practically tied with SwinT, coming only marginally behind but still ahead of the worst case ViT. As we've seen before, training accuracy is not the only metric we have for comparison and this trend may not be consistent.

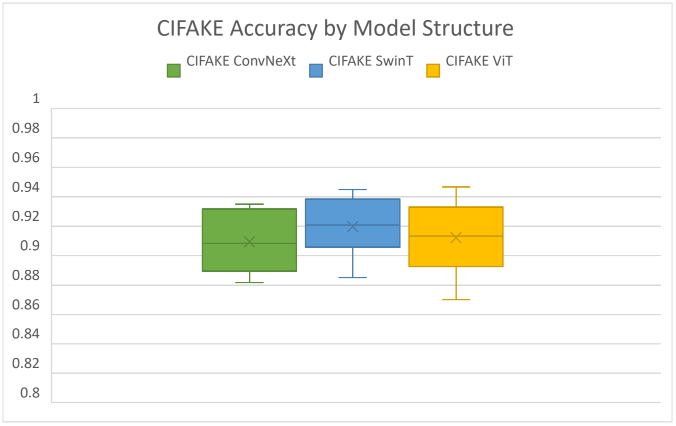

Fig. 12.

Finally we take the strongest performing hyper-parameters yet again which are still a learning rate and batch size equal to $LR = 1e-3$, and $BS = 64$ respectively. Following the retraining, during the 25 epochs we observe their relative validation losses. Incredibly, given the rocky start of training accuracy, we have a loss graph (Fig. 13) that shows a completely different story. Despite the fact that on average and in best case scenarios, the ConvNeXt model scored lower than both the transformers, its losses are leading considerably in both stability as well as the loss itself. SwinT also scores better than ViT as we'd expect. This is an indication of how

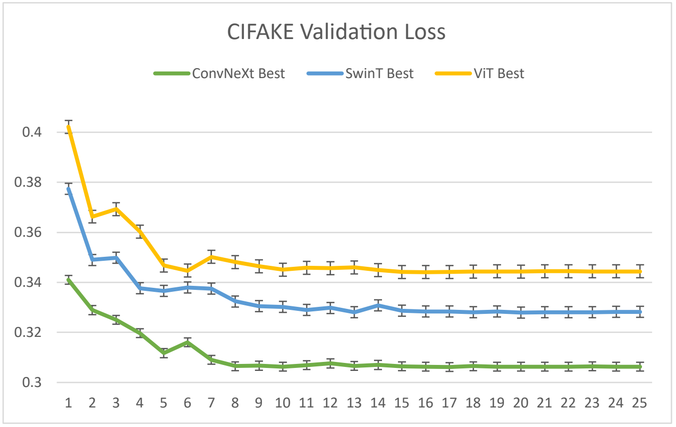

Fig. 13.

accuracy cannot be the only metric evaluated regarding a models' performance. We now will observe the relative loss functions on the test data. Thus we now begin judging on unseen data performance. In conjunction with validation loss shown by Fig. 14, we now observe that the loss metrics for all three functions are what we expect. ConvNeXt has lead over SwinT in both Label Smoothed Cross Entropy Loss and the base Cross Entropy Loss. This is a strong indicator that while the training accuracies of these models may have been close, the models behave in an expected way when we observe how closely they managed to predict the ground truth. Lastly, we observe test accuracy, f1 score, precision and recall: Across all four metrics in Fig. 15, we observe that once again

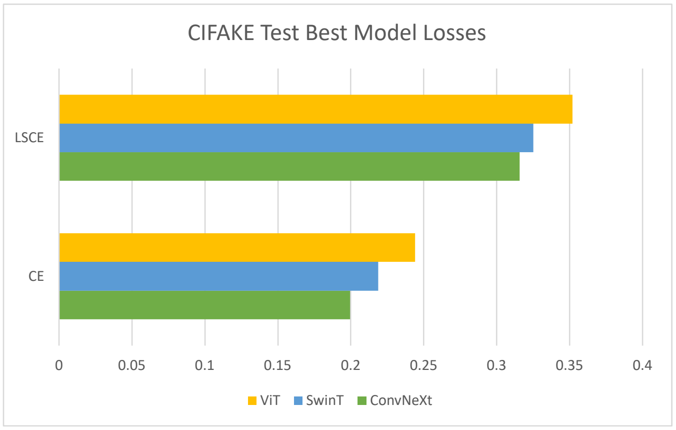

Fig. 14.

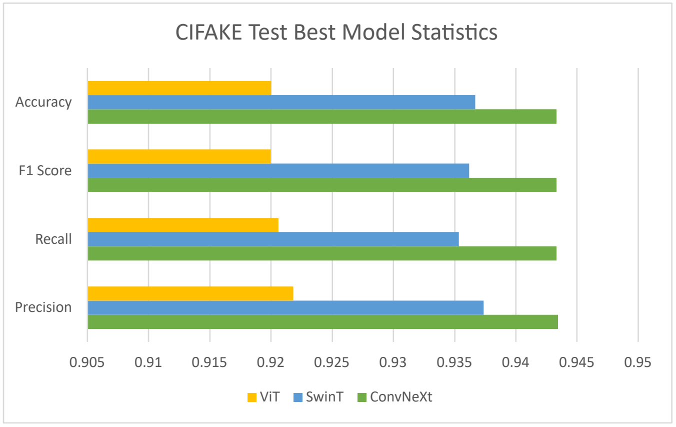

Fig. 15.

ConvNeXt dominates ViT and maintains a healthy lead above SwinT. Given the class balance, we note the slight gains on precision over recall that is present in both transformers while ConvNeXt maintains a perfectly balanced accuracy, F1 score, precision and recall. This is an indicator of confidence that will be anaylsed later in results.

Despite the training accuracy being deceptive; all other metrics served to prove the hierarchy that we expect from the architectures. ConvNeXt once again being dominant with SwinT shortly behind while ViT lags behind.

E. Limitations

The data-collection and execution components of this report were conducted under certain constraints that require acknowledgement before any conclusions can be accurately understood with respect to context. The constraints of the system on which all training and testing was carried out, prevented the exploration of sub-optimal methods with prolonged convergence periods and pre-trained models of the outlined architectures that had a memory demand which greatly exceeded the base model. The system comprised of a singular RTX 2070 SUPER graphics card. Despite the fact that this is a relatively powerful card, the constraining factor remained the Dedicated GPU Memory. When handling large models, batch sizes and image resolutions, the Dedicated GPU Memory becomes overwhelmed, resulting in substantial performance degradation, in some cases doubling the convergence period or worse.

Consequently, we confined our models to their base cases. All input tensors adhered to the standard ImageNet 21k [16] resolutions of 224×224, corresponding to the pre-training input size of any of the given models. Furthermore, due to the limitations [IV,E] with regards to time constraints, we refrained from optimizing Optimizers, Schedulers, and other minor adjustments per model for all three of the datasets. We instead opted to keep these as stable as possible to provide a reasonable comparison between all architectures without heavily optimizing any per chance by comparison. That being said, we made a concerted effort to ensure that the most widely optimal choices for these parameters were made and maintained consistently across models, again to minimize bias in the final evaluation and conclusions.

While there were a total of 72 permutations of models that were tested, it should be noted that the testing was not exhaustive. This limitation should be kept in mind when interpreting the conclusions drawn from this research as there may exist better hyper-parameters per architecture which can in turn lead to higher accuracy or lower loss across the board, but from our testing we believe we found parameters that provided no egregious bias to any of the architectures which, in turn, allowed relatively fair comparisons.

V. Conclusion

In this study we gathered comprehensive evidence toward evaluating the state-of-the-art Image Classification architectures. Through applications of transfer learning and feature detection using pre-trained base models for each architecture, we have found that ConvNeXt outperforms both transformers by a large margin across all metrics as a general backbone architecture for image classification. Additionally, SwinT performs better than or equal to that of the base ViT but in general performs well above the latter across all metrics.

Training of each of these models was relatively straight forward given they are all pre-trained on ImageNet 21k [27]/ and required no extensive high end machinery past what was outlined in the limitations section of this report.

Their model sizes as exported by Pytorch were as follows: Any given binary-classification ConvNeXt base model occupied 314,198 KB. Slightly larger was the SwinT binary-classification base model which occupied 338,978KB and finally the ViT binary-classification base model occupied 335,215KB. There is very little difference in size, but it is interesting to note that the most space efficient model is indeed the most recent one and the only convolutional neural network, ConvNeXt.

A comparison of each of their 25-epoch training runs across all three datasets resulted in ConvNeXt achieving a full training time of $\sim$ 89.59 minutes. SwinT took slightly longer at $\sim$ 92.58 minutes and ViT took the least amount of time, taking only $\sim$ 79.62 minutes to train. We expected ViT to be faster given the low resolutions of the datasets, but it is interesting to observe that despite the scaling optimizations of SwinT, ConvNeXt managed to be marginally quicker.

As to be expected, ConvNeXt provided the predominant architecture. With F1 scores being greater than that of both transformers across each domain. However it was interesting to note the performance degradation that occurred during training over the CIFAKE dataset. While it's not absolutely evident as to why in this case, the Vit and SwinT alike had such dominant general case performance initially, we quickly saw ConvNeXt return to dominance when applying the best hyper-parameters over a test dataset. Examination of the Validation Loss (Figures 6,10,14) over 25-epochs showed that ConvNeXt was once again much closer to the ground truth than either of its Transformer counterparts.

A. Quantitative Analysis

TABLE I
Testing Results across Best Models

	Precision	**Recall**	**F1 Score**	**Accuracy**
LANDUSE				
ConvNeXt Best	0.988886	0.988571	0.988566	0.988571
SwinT Best	0.98324	0.982857	0.982758	0.982857
ViT Best	0.966434	0.965714	0.965797	0.965714
XRAY				
ConvNeXt Best	0.962364	0.945917	0.953646	0.962199
SwinT Best	0.929611	0.929611	0.929611	0.941581
ViT Best	0.933595	0.924489	0.928873	0.941581
CIFAKE				
ConvNeXt Best	0.943429	0.94331	0.943328	0.943333
SwinT Best	0.937374	0.935341	0.936187	0.936667
ViT Best	0.921827	0.920589	0.919968	0.92

Quantitatively, with reference to Table I, a blatant hierarchy of models was established. ConvNeXt provided a dominant showing with F1 scores $= [0.989, 0.954, 0.943]$ for the Land Use, X-Ray, and CIFAKE datasets respectively.

The SwinT architecture was not far behind only being less by $-0.588, -2.520, -0.757$ percent across all three datasets. Other than the X-Ray dataset, it is clear to see ConvNeXt, in what we assume to be the general case, has only marginal gains over the SwinT architecture. Those gains could be improved, given theoretically optimal hyper-parameter tuning, but even without it still showcases the power discrepancy.

Lastly we see the base ViT performing poorly across all metrics. Against ConvNeXt its F1 scores lag behind by $-2.303, -2.598, -2.476$ percentage points across all the datasets respectively. This was to be expected given that each of these models were built off the back of the last, and the ViT being the first successful implementation of transformers in Computer Vision.

B. Additional Information

A more in depth view of training performances can be observed in the Appendix. Furthermore, many more graphs and all of the collected data are given on the Github repository under 'Data.xlsx'.

C. GitHub

The code for this report, extra graphs, and statistical evidence is available on the Github Repository.

VI. Additional References

It is important that we aknowledge some code repositories and/or articles that helped in the creation of datasets and/or models. Other than those which have already been mentioned: A considerable amount was learnt from the ConvNeXt Github page [1]. Elements of dataset handling and data augmentation was learnt from 'CIFAR-100 ConvNeXt' by 'mtamer1418' on Kaggle [9]. All pre-trained models were obtained from the python library 'timm' [26], created by Ross Wightman [29]. 'Shreydan' on Kaggle provided insight into the timm library [10]. Lastly a better understanding of the SwinT architecture in pytorch was provided by the official Swin Transformer Github repository [32], as well as 'pdochannel' on Kaggle [33].

References

[1] *A ConvNet for the 2020s*. original-date: 2022-01-06T00:53:24Z. Nov. 2023. URL: https://github.com/facebookresearch/ConvNeXt (visited on 11/05/2023).

[2] Chinmay Bhalerao. *Vision Transformers [ViT]: A very basic introduction*. en. June 2023. URL: https://medium.com/data-and-beyond/vision-transformers-vit-a-very-basic-introduction-6cd29a7e56f3 (visited on 11/08/2023).

[3] *Bilinear Interpolation - an overview — ScienceDirect Topics*. URL: https://www.sciencedirect.com/topics/engineering/bilinear-interpolation (visited on 11/07/2023).

[4] Jordan J. Bird and Ahmad Lotfi. *CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images*. arXiv:2303.14126 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2303.14126. URL: http://arxiv.org/abs/2303.14126 (visited on 11/05/2023).

[5] Shenglong Chen, Yoshiki Ogawa, and Yoshihide Sekimoto. "Large-scale individual building extraction from open-source satellite imagery via super-resolution-based instance segmentation approach". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 195 (Jan. 2023), pp. 129–152. DOI: 10.1016/j.isprsjprs.2022.11.006.

[6] *CIFAKE: Real and AI-Generated Synthetic Images*. en. URL: https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images (visited on 11/05/2023).

[7] *CIFAR-10*. en. Page Version ID: 1166845013. July 2023. URL: https://en.wikipedia.org/w/index.php?title=CIFAR-10&oldid=1166845013 (visited on 11/07/2023).

[8] *CIFAR-10 and CIFAR-100 datasets*. URL: https://www.cs.toronto.edu/~kriz/cifar.html (visited on 11/07/2023).

[9] *CIFAR-100 ConvNeXT*. en. URL: https://kaggle.com/code/mtamer1418/cifar-100-convnext (visited on 11/05/2023).

[10] *ConvNeXt-Tiny — lightning + timm*. en. URL: https://kaggle.com/code/shreydan/convnext-tiny-lightning-timm (visited on 11/05/2023).

[11] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs]. June 2021. URL: http://arxiv.org/abs/2010.11929 (visited on 11/05/2023).

[12] "Fake Trump arrest photos: How to spot an AI-generated image". en-GB. In: *BBC News* (Mar. 2023). URL: https://www.bbc.com/news/world-us-canada-65069316 (visited on 11/07/2023).

[13] Manav Garg et al. "Comparative Analysis of Deep Learning Architectures and Vision Transformers for Musical Key Estimation". en. In: *Information* 14.10 (Oct. 2023). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 527. ISSN: 2078-2489. DOI: 10.3390/info14100527. URL: https://www.mdpi.com/2078-2489/14/10/527 (visited on 11/06/2023).

[14] Kalley Huang. "Why Pope Francis Is the Star of A.I.-Generated Photos". en-US. In: *The New York Times* (Apr. 2023). ISSN: 0362-4331. URL: https://www.nytimes.com/2023/04/08/technology/ai-photos-pope-francis.html (visited on 11/07/2023).

[15] Nghi Huynh. *Understanding Loss Functions for Classification*. en. Mar. 2023. URL: https://medium.com/mlearning-ai/understanding-loss-functions-for-classification-81c19ee72c2a (visited on 11/06/2023).

[16] *ImageNet*. URL: https://www.image-net.org/index.php (visited on 11/05/2023).

[17] *keremberke/chest-xray-classification · Datasets at Hugging Face*. Feb. 2023. URL: https://huggingface.co/datasets/keremberke/chest-xray-classification (visited on 11/05/2023).

[18] Daniel S. Kermany et al. "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning". English. In: *Cell* 172.5 (Feb. 2018). Publisher: Elsevier, 1122–1131.e9. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2018.02.010. URL: https:

//www.cell.com/cell/abstract/S0092-8674(18)30154-5 (visited on 11/05/2023).

[19] Alex Krizhevsky. “Convolutional Deep Belief Networks on CIFAR-10”. en. In: ().

[20] *Land-Use Scene Classification*. URL: https://www.kaggle.com/datasets/apollo2506/landuse-scene-classification?select=images (visited on 11/05/2023).

[21] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. arXiv:2103.14030 [cs]. Aug. 2021. DOI: 10.48550/arXiv.2103.14030. URL: http://arxiv.org/abs/2103.14030 (visited on 11/05/2023).

[22] Zhuang Liu et al. *A ConvNet for the 2020s*. arXiv:2201.03545 [cs]. Mar. 2022. URL: http://arxiv.org/abs/2201.03545 (visited on 11/05/2023).

[23] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. arXiv:1711.05101 [cs, math] version: 3. Jan. 2019. DOI: 10.48550/arXiv.1711.05101. URL: http://arxiv.org/abs/1711.05101 (visited on 11/07/2023).

[24] *Loss Functions — timmdocs*. URL: https://timm.fast.ai/loss.cross_entropy (visited on 11/06/2023).

[25] *NLLLoss — PyTorch 2.1 documentation*. URL: https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html (visited on 11/06/2023).

[26] *Pytorch Image Models (timm) — timmdocs*. URL: https://timm.fast.ai/ (visited on 11/09/2023).

[27] Tal Ridnik et al. *ImageNet-21K Pretraining for the Masses*. arXiv:2104.10972 [cs]. Aug. 2021. DOI: 10.48550/arXiv.2104.10972. URL: http://arxiv.org/abs/2104.10972 (visited on 11/05/2023).

[28] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. arXiv:2112.10752 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2112.10752. URL: http://arxiv.org/abs/2112.10752 (visited on 11/07/2023).

[29] *Ross Wightman (@wightmanr) / X*. en-GB. June 2022. URL: https://twitter.com/wightmanr (visited on 11/09/2023).

[30] *StepLR — PyTorch 2.1 documentation*. URL: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html (visited on 11/07/2023).

[31] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. “One pixel attack for fooling deep neural networks”. In: *IEEE Transactions on Evolutionary Computation* 23.5 (Oct. 2019). arXiv:1710.08864 [cs, stat], pp. 828–841. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2019.2890858. URL: http://arxiv.org/abs/1710.08864 (visited on 11/07/2023).

[32] *Swin Transformer*. original-date: 2021-03-25T12:42:36Z. Nov. 2023. URL: https://github.com/microsoft/Swin-Transformer (visited on 11/05/2023).

[33] *Swin Transformer in PyTorch*. en. URL: https://kaggle.com/code/pdochannel/swin-transformer-in-pytorch (visited on 11/05/2023).

[34] *UC Merced Land Use Dataset*. URL: http://weegee.vision.ucmerced.edu/datasets/landuse.html (visited on 11/05/2023).

[35] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. URL: http://arxiv.org/abs/1706.03762 (visited on 11/08/2023).

[36] Yi Yang and Shawn Newsam. “Bag-of-visual-words and spatial extensions for land-use classification”. en. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. San Jose California: ACM, Nov. 2010, pp. 270–279. ISBN: 978-1-4503-0428-3. DOI: 10.1145/1869790.1869829. URL: https://dl.acm.org/doi/10.1145/1869790.1869829 (visited on 11/05/2023).

Appendix

TABLE II
Top Model Accuracy and Loss for ConvNeXt

ConvNeXt$_{5,1e-3,64}$	1	2	3	4	5
LANDUSE					
train_loss	1.0792	0.7771	0.7327	0.7102	0.6976
train_acc	0.8626	0.9709	0.9837	0.9895	0.9924
val_loss	0.8247	0.7709	0.7498	0.737	0.7297
val_acc	0.9524	0.9705	0.9771	0.9781	0.9814
XRAY					
train_loss	0.4623	0.3432	0.3257	0.3138	0.3077
train_acc	0.8322	0.9205	0.9303	0.9392	0.9446
val_loss	0.3517	0.3143	0.3077	0.2961	0.2874
val_acc	0.903	0.9433	0.9459	0.9536	0.9631
CIFAKE					
train_loss	0.4191	0.3505	0.3338	0.3279	0.3225
train_acc	0.8574	0.9129	0.9261	0.9267	0.9335
val_loss	0.3484	0.3305	0.3324	0.3292	0.3191
val_acc	0.9221	0.9286	0.9286	0.9236	0.9336

TABLE III
ACCURACY AND LOSS FOR TRAINING AND VALIDATION ACROSS ALL DATASETS

ConvNeXt$_{10,1e-3,64}$	1	2	3	4	5	6	7	8	9	10
LANDUSE										
train_loss	1.0776	0.7741	0.7316	0.7111	0.6964	0.6876	0.6798	0.6676	0.6663	0.6657
train_acc	0.8585	0.9735	0.9854	0.9901	0.994	0.9958	0.9966	0.9978	0.9985	0.9982
val_loss	0.8279	0.782	0.7493	0.7363	0.7324	0.7254	0.7186	0.7156	0.7158	0.7153
val_acc	0.9552	0.9667	0.9786	0.9829	0.9805	0.9871	0.9857	0.9895	0.9871	0.9876
XRAY										
train_loss	0.4171	0.3314	0.3179	0.31	0.3098	0.3041	0.303	0.2953	0.2941	0.294
train_acc	0.8629	0.9272	0.9375	0.9424	0.9458	0.949	0.9482	0.9546	0.9571	0.9566
val_loss	0.3337	0.3096	0.3071	0.2891	0.2958	0.2836	0.291	0.2795	0.2822	0.2802
val_acc	0.9356	0.9476	0.9425	0.9639	0.9494	0.9648	0.9597	0.9665	0.9648	0.9648
CIFAKE										
train_loss	0.4109	0.3453	0.3332	0.3268	0.3243	0.3242	0.3185	0.3105	0.3081	0.3097
train_acc	0.8641	0.9178	0.9258	0.9352	0.934	0.9319	0.9412	0.9428	0.9472	0.9476
val_loss	0.3453	0.3407	0.3255	0.3242	0.3224	0.3233	0.3201	0.3164	0.3081	0.3097
val_acc	0.9193	0.915	0.9314	0.93	0.9364	0.9279	0.9336	0.9386	0.9393	0.9393

TABLE IV
ACCURACY AND LOSS FOR TRAINING AND VALIDATION ACROSS ALL DATASETS

SwinT$_{5,1e-3,64}$	1	2	3	4	5
LANDUSE					
train_loss	1.1968	0.8262	0.7799	0.7584	0.742
train_acc	0.8454	0.9582	0.9732	0.9778	0.9841
val_loss	0.8577	0.7993	0.778	0.7627	0.7521
val_acc	0.9476	0.9662	0.9714	0.9733	0.9767
XRAY					
train_loss	0.4805	0.3748	0.346	0.3393	0.3298
train_acc	0.8178	0.8992	0.9144	0.9272	0.9299
val_loss	0.3594	0.3288	0.3151	0.3065	0.3042
val_acc	0.9013	0.9348	0.9425	0.9536	0.9425
CIFAKE					
train_loss	0.4257	0.3641	0.3515	0.3443	0.3424
train_acc	0.8588	0.9064	0.915	0.9218	0.9226
val_loss	0.3785	0.3527	0.3417	0.3438	0.3399
val_acc	0.8936	0.9193	0.9257	0.9243	0.9307

TABLE V
ACCURACY AND LOSS FOR TRAINING AND VALIDATION ACROSS ALL DATASETS

$\text{SwinT}_{10,1e-3,64}$	1	2	3	4	5	6	7	8	9	10
LANDUSE										
train_loss	1.2091	0.8292	0.7832	0.7597	0.7443	0.7327	0.7245	0.713	0.7137	0.7119
train_acc	0.8456	0.9588	0.9705	0.9774	0.9814	0.9863	0.9888	0.9921	0.991	0.992
val_loss	0.8589	0.8086	0.7784	0.765	0.7541	0.7543	0.7441	0.7385	0.7376	0.7368
val_acc	0.9476	0.959	0.97	0.9757	0.9776	0.979	0.9767	0.981	0.981	0.9805
XRAY										
train_loss	0.4814	0.3763	0.3486	0.3343	0.3257	0.3228	0.3186	0.3147	0.3138	0.3102
train_acc	0.8168	0.8918	0.9149	0.9269	0.9357	0.9338	0.9372	0.9406	0.9438	0.9438
val_loss	0.361	0.3282	0.3179	0.3063	0.3003	0.2969	0.2941	0.2949	0.2947	0.2969
val_acc	0.9056	0.9279	0.9459	0.9502	0.9562	0.9622	0.9648	0.9657	0.9657	0.9639
CIFAKE										
train_loss	0.4319	0.3649	0.3539	0.346	0.3443	0.3417	0.3405	0.3316	0.3313	0.3338
train_acc	0.856	0.9073	0.9152	0.9187	0.9202	0.9248	0.9216	0.9297	0.9317	0.93
val_loss	0.3748	0.3535	0.3458	0.3515	0.341	0.359	0.3325	0.3304	0.3317	0.3303
val_acc	0.9014	0.9171	0.9229	0.9143	0.925	0.92	0.9293	0.9286	0.9293	0.93

TABLE VI
ACCURACY AND LOSS FOR TRAINING AND VALIDATION ACROSS ALL DATASETS

$\text{ViT}_{5,1e-3,64}$	1	2	3	4	5
LANDUSE					
train_loss	1.1536	0.8051	0.762	0.736	0.7221
train_acc	0.8331	0.9717	0.9834	0.991	0.9944
val_loss	0.8796	0.826	0.8097	0.7907	0.7887
val_acc	0.9423	0.9576	0.9614	0.968	0.9652
XRAY					
train_loss	0.4258	0.3379	0.3214	0.3127	0.3157
train_acc	0.8471	0.9237	0.9357	0.9423	0.944
val_loss	0.3332	0.31	0.3078	0.3026	0.2957
val_acc	0.9347	0.9476	0.9502	0.9433	0.9579
CIFAKE					
train_loss	0.4486	0.3715	0.3553	0.3499	0.3417
train_acc	0.8324	0.8999	0.9073	0.9149	0.9216
val_loss	0.3865	0.3648	0.3594	0.3498	0.3556
val_acc	0.8907	0.9057	0.9035	0.9157	0.915

TABLE VII
ACCURACY AND LOSS FOR TRAINING AND VALIDATION ACROSS ALL DATASETS

$\text{ViT}_{10,1e-3,64}$	1	2	3	4	5	6	7	8	9	10
LANDUSE										
train_loss	1.1577	0.8112	0.7638	0.7389	0.7227	0.7116	0.7046	0.6901	0.6875	0.6865
train_acc	0.8295	0.9678	0.9831	0.9905	0.9933	0.9961	0.9969	0.9988	0.9988	0.999
val_loss	0.8864	0.8312	0.8111	0.7932	0.7827	0.7813	0.7764	0.7698	0.7694	0.7692
val_acc	0.9338	0.951	0.9619	0.9676	0.9676	0.9705	0.9695	0.9705	0.9714	0.9719
XRAY										
train_loss	0.4554	0.3433	0.328	0.3261	0.313	0.311	0.3105	0.3043	0.3	0.3013
train_acc	0.8366	0.923	0.9318	0.9343	0.9455	0.9429	0.9419	0.9463	0.9512	0.9517
val_loss	0.3528	0.3189	0.3135	0.2987	0.296	0.2962	0.295	0.289	0.2899	0.2884
val_acc	0.9056	0.9416	0.9425	0.9528	0.9588	0.9519	0.9562	0.9631	0.9639	0.9614
CIFAKE										
train_loss	0.449	0.3779	0.359	0.3516	0.3514	0.3466	0.3464	0.3332	0.3358	0.333
train_acc	0.8355	0.8934	0.9081	0.9121	0.9123	0.9139	0.9171	0.9267	0.9247	0.9267
val_loss	0.3717	0.3642	0.3484	0.3568	0.3444	0.3333	0.3426	0.3305	0.331	0.3304
val_acc	0.8914	0.9079	0.9136	0.9107	0.9121	0.9271	0.9171	0.9343	0.9286	0.9314