Exploring Deep Learning Methods for Head CT Triaging

Jonathan Coronado
Stanford University
Stanford, CA
jonathan.coronado@stanford.edu

Grant Sheen
Stanford University
Stanford, CA
gsheen@stanford.edu

Abstract

In this project, we explore two deep learning methods for automating the process of head CT (HCT) scan triage. The first proposed method is a mini U-Net, based on an architecture that is specialized in medical image segmentation. We also propose the use of a Data-efficient Image Transformer (DeiT), a computationally efficient transformer model. Though we encountered tough computational and memory constraints, our experiments demonstrated the potential of these architectures for HCT triage. The mini U-Net achieved an accuracy of 0.6509 on a subset of 400 test examples, while the DeiT model achieved a validation accuracy of 54%. Our findings indicate that, with sufficient computational resources, specialized architectures like U-Net could outperform traditional Dense CNNs in medical image analysis, particularly for tasks involving large and complex datasets such as HCT scans. Further research is needed to optimize these models and fully realize their potential in clinical settings.

1. Introduction

Medical image triaging is the process of prioritizing the review and analysis of medical images, such as head CT (HCT) scans, based on the urgency of the patient's condition. This process ensures that the most critical cases, which require immediate attention and intervention, are identified and addressed first. Triaging involves evaluating images to detect signs of severe or life-threatening conditions, such as internal bleeding, fractures, or tumors, and then ranking these cases in order of priority. The goal of medical image triaging is to optimize workflow efficiency, reduce the time to diagnosis, and improve patient outcomes by ensuring that urgent cases receive prompt and appropriate medical attention.

Triaging head CT scans has historically requires manual review by radiologists but faces challenges such as high data volume and radiologist shortages. Machine learning models for head CT triaging is increasingly becoming integrated with modern radiology practices. The most commonly used tools by radiologists are Rapid AI and Viz.ai. These tools use convolutional neural networks (CNN) trained on large datasets to detect various types of intracranial hemorrhages and alert radiologists to critical findings in real-time. Developing accurate triaging models has the potential to enhance diagnostic accuracy, efficiency, and consistency, ultimately improving the efficacy of patient care.

For our project, we wanted to explore various deep learning approaches towards HCT triaging. Our model starts with a head CT scan as an input. We experimented with either using the raw sinogram or performing our custom preprocessing method. We then trained both a U-Net and a data-efficient image transformer (DeiT) to output a predicted label of either normal or abnormal. By comparing the performance of our U-Net model, the DeiT model, and existing benchmarks, we aim to understand the efficacy of various deep learning architectures in HCT triaging.

2. Related Work

Our project was inspired and informed by numerous pioneering projects in the biomedical imaging and computer vision space.

2.1. Impact of Upstream Medical Image Processing on Downstream Performance of a Head CT Triage Neural Network

This paper was the direct inspiration for our project because they created the dataset that we are using and are the only existing published paper that uses the dataset. As a result, it is also the benchmark we will use to compare our results.

It explored various processing steps of head CT (HCT) scans before passing them into a CNN. They experimented with training a CNN on raw sinogram data, varying the number of x-ray projections used on images, and using CT windowing for preprocessing. They were able to achieve a mean area under the receiver operating characteristic curve of 0.84 at triaging head CT studies as normal or abnormal.

There were two primary takeaways from this paper: 1) Training a triage CNN directly on sinograms resulted in comparable performance to inputting reconstructed head CT studies, which suggests that models can be positioned further upstream. 2) CT windowing was the most effective form of pre-processing as it achieved their highest AUC. [1]

Our project builds upon their research by training our own U-Net model farther upstream and incorporating CT windowing into our preprocessing pipeline.

2.2. U-Net: Convolutional Networks for Biomedical Image Segmentation

This paper explores the possibility of a neural network architecture that is specialized for biomedical image segmentation and able to perform well with a smaller training dataset. Their final product was U-Net, an award winning CNN architecture that was later trained to win the 2015 ISBI cell tracking challenge [4].

This paper is of interest to our work because it presents a CNN architecture that is specialized for biomedical imaging. Such an architecture can be used to build upon the work of the previously discussed paper, which did not use a specialized biomedical imaging architecture. Further, since the previous paper showed no statistically significant decrease in performance when trained on raw data, there is potential for the U-Net architecture to show performance improvements when trained on raw sinogram data as well.

As a result, our plan is to train a U-Net classification model directly on the HCT sinograms and evaluate its performance against the original paper.

2.3. Training data-efficient image transformers & distillation through attention

This paper introduces the Data-efficient Image Transformer (DeiT), which is a compact image transformer model designed to have significantly less computational requirements. This work is pioneering because it demonstrates that competitive convolution-free transformers can be trained on the ImageNet dataset using a single computer in less than three days. The DeiT model, with 86 million parameters, achieves a top-1 accuracy of 83.1% on ImageNet with no external data. The authors also introduce a novel teacher-student strategy specific to transformers, utilizing a distillation token that ensures the student model learns from the teacher through attention. [7]

This work is relevant to our project because it allows us to experiment with a transformer architecture given our limited computational resources. Vision Transformers have recently shown superior performance on large-scale image classification tasks such as ImageNet compared to CNNs. As a result, DeiT presents as an alluring alternative to CNN-based architectures that could have more robust capabilities. Given the performance of DeiT in other image classification domains, we hypothesize that it could also perform well in the context of HCT triaging. In particular, the inclusion of the attention mechanism could help the model focus on the most relevant parts of the image.

2.4. MONeT: Memory Optimization for Deep Networks

This paper presents a framework for deep learning that optimizes deep learning by reducing memory usage with minimal computational overhead. Interestingly, this paper described the use of this framework with U-Nets, which are memory and computationally intensive models. Overall, MONeT has been shown to be capable of reducing memory requirements by up to 3x for various PyTorch models while only incurring a computational overhead of 9-16% [6].

This work is relevant to our project because it showcases various methods of memory optimizations which were necessary in order to get our model to run. Without these methods, our model quickly runs out of CUDA memory due to the intense requirements of U-Nets.

2.5. Dense Convolutional Network and Its Application in Medical Image Analysis

In this paper, a dense convolutional network (Dense CNN) is proposed and shown to be effective for medical image analysis. A Dense CNN architecture is one where each layer is connected to each layer, much like neurons in a Multi-Layer Perceptron. These networks work well for medical image analysis due to the dense connections between layers, which allow for feature reuse and counteract vanishing gradients [9].

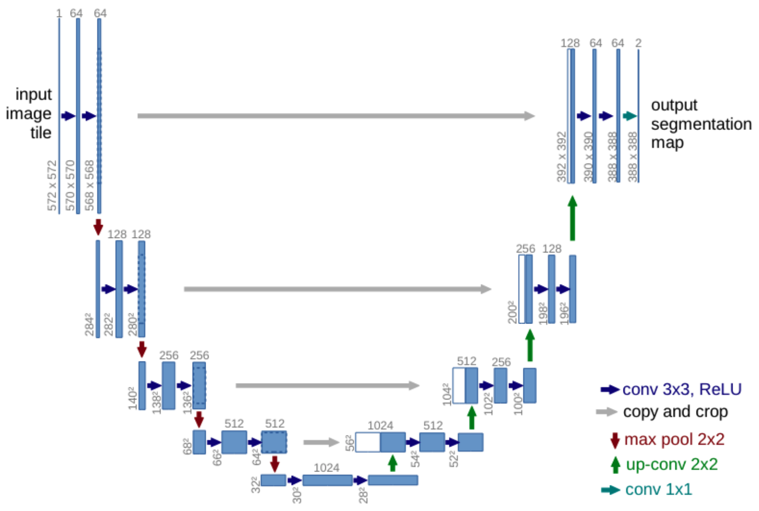
This paper is relevant to our project because it was the original network used for the project ours is based on [1]. Dense CNNs are effective at capturing detailed relationships and features in medical imaging, but generally unspecialized. We posit that with an architecture specialized for medical imaging, it is possible to outperform the original achieved by Hooper et al.

3. Methods

3.1. Mini U-Net

Our project makes use of a U-Net, an architecture for a CNN that is specialized for medical image processing [4]. The U-Net architecture described by Ronneberger et al consists of four encoder (convolutional downsampling) blocks, followed by a bottleneck convolutional block, then four convolutional decoder (convolutional upsampling) blocks. Each encoder block consists of two unpadded 3x3 convolutions, each followed by a ReLU and finally a 2x2 maxpool with stride 2. Each encoding block doubles the number of feature channels as the network contracts. Each decoder block consists of the same two unpadded convolutions and

ReLUs, this time followed by a 2x2 upsampling operation which doubles the number of feature channels, then a concatenation with the feature map from the corresponding encoder block. This feature map is cropped due to the loss of border pixels in each convolution. Ronneberger et al then adds a 1x1 convolution to map each feature vector to the desired number of classes. The original U-Net architecture is shown below.

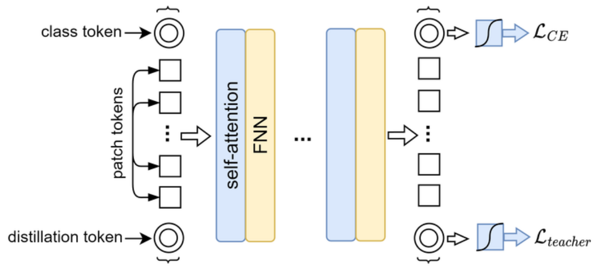


Our model made slight adjustments to the architecture originally proposed by Ronneberger et al, which was intended for medical image segmentation rather than binary classification. Rather than transition from the final decoder block directly to a convolution and feature map, we added additional convolution block followed by a global average pool to compress our spatial information into one class.

3.2. Transformer

We initially tried building our own custom vision transformer model. However, once we started training it, we ran into GPU memory issues since the model was too large and even a batch size of 1 was not possible. As a result, we looked into other alternatives and found DeiT, which was computationally feasible.

We explored finetuning the DeiT model for HCT triaging. The DeiT model employs a teacher-student distillation approach, which enhances its performance by transferring knowledge through a distillation token.



The DeiT model was initialized with pre-trained weights from training on the ImageNet dataset, providing a strong starting point with robust feature extraction capabilities.

3.3. Memory Optimizations

Our project was subject to memory constraints that, for the size of our inputs and model, were tough to meet. As such, several memory optimizations were implemented. First, in order to provide some normalization, we initially intended to use Batch Normalization, but quickly ran out of CUDA memory when using a batch size larger than one [2]. To remedy this, we instead opted for Instance Normalization between each convolution, which normalizes the mean and variance of each individual sample rather than across an entire batch [8]. This is equivalent to Group Normalization, but with a minibatch size of 1. Also, we used gradient accumulation to achieve a ”virtual batch” size of 2 before backpropagating gradients. We used this relatively small accumulation step size to balance only being able to train over one epoch. Further, we were required to reduce the number of feature channels in each layer by a factor of 4 from Ronneberger et al’s original implementation in order to complete even one forward pass. Additionally, we implemented several common memory optimizations, such as gradient checkpointing, mixed precision, and gradient scaling [6]. Finally, to meet our memory constraints, we had to reduce the number of feature channels at each layer by a factor of four. Even so, we had a total of 5648305 learnable parameters.

4. Dataset and Features

Our model is trained on a subset of the SinoCT dataset, which contains over 9,000 HCT scans, each labeled as normal or abnormal. Each scan within the dataset contains a reconstructed image and a corresponding sinogram. The reconstructed images are 512x512 pixels with a variable number of axial slices per scan, typically between 40 and 50. The sinograms are 984x888 pixels with a variable number of axial slices per scan. The full dataset is 1.3TB. The reading radiologist designated each CT scan as normal or abnormal at the time of original image interpretation as part of standard clinical procedure. Of 9776 total scans, 5398

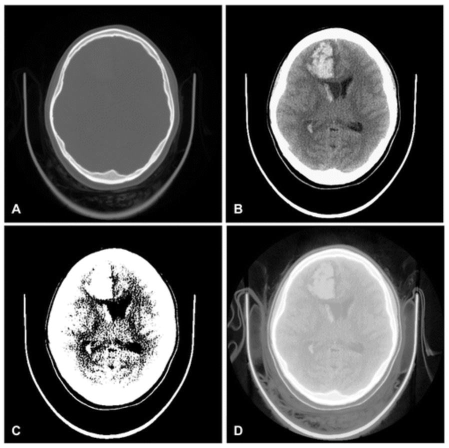


Figure 1. The same axial section of a noncontrast head CT image with different preprocessing operations: (A) original image, (B) image with a blood CT window applied, (C) image with a stroke window applied, and (D) histogram-equalized image.

(55.21%) were abnormal. For our experiments, we used a split of 80/20 for training/testing/validation sets. This gave us 7820 training examples and 1956 testing examples.

4.1. Pre-Processing

We developed our own pre-processing pipeline for the HCT sinograms in order to reduce the dimensionality of our data to fit the DeiT model. The SinoCT dataset provides around 40 .dcm files that each correspond to a slice of a sinogram. Our pipeline started with performing CT windowing on each of the files.

We did our own implementation of blood, stroke, and histogram-equalized windowing. Here is an example of the CT windowed images, which was borrowed from Hooper [1].

These CT windowed images are then stacked along the depth dimension to create 3D tensors for each channel. The script averages the slices along the depth dimension to produce 2D images for each channel, which are then combined to form a single 3-channel image.

4.2. Data Loading

When loading the data we had to make two modifications. First, we needed to transpose the data to the correct dimensions. The original data was in the form [H, W, D] and transposed to [1, D, H, W] as expected by pytorch. Here, D is the number of axial slices, and we add a dimension to account for there being one channel per slice. Next, we found that after training for a while, certain examples would be corrupted and appear as None or malformed data in the set. To remedy this, error handling was added to skip over corrupt examples. This was only used a handful of times as this corrupt examples appeared very sparsely.

5. Experiments

We experiment with two approaches to automating HCT triage. In one experiment, we use a mini U-Net tested and trained on a subset of the raw sinogram data with instance normalization between each convolution. In another experiment, we use a DeiT model to train and test on reconstructed images.

5.1. Training a mini U-Net on Sinograms

For the first experiment, we use a mini U-Net, as described in 3.1. Due to memory constraints, we were not able to train over the entire training set. Rather, training was stopped after 4003 iterations, which took around three hours. Additionally, to account for a small effective training set, our model was only tested on 400 examples from the test set. Our model used Adam optimization and Binary Cross Entropy (BCE) Loss [3][5]. Both of these design choices were made in line with the original paper [1]. Default PyTorch hyperparameters were used, including a starting learning rate of 1e-3.

Aside from accuracy, we used several evaluation metrics for this experiment. One such metric was precision, which is defined as a measure of $\frac{\text{\# correct positive predictions}}{\text{\# total positive predictions}}$ and measures the accuracy of positive predictions. We also used recall, which is defined as $\frac{\text{\# correct positive predictions}}{\text{\# total positive examples}}$ and can be thought of as the ability of a model to identify positive examples. Another metric we used was F1 score, which is $2 * \frac{\text{precision} \cdot \text{recall}}{\text{precision + recall}}$ and is important when false positives and negatives mean different things. Finally, we also used Area Under the Receiver Operating Characteristic (AUROC) which measures the ability of a model to find positive examples without classifying false positives. The baseline for this value is 0.5.

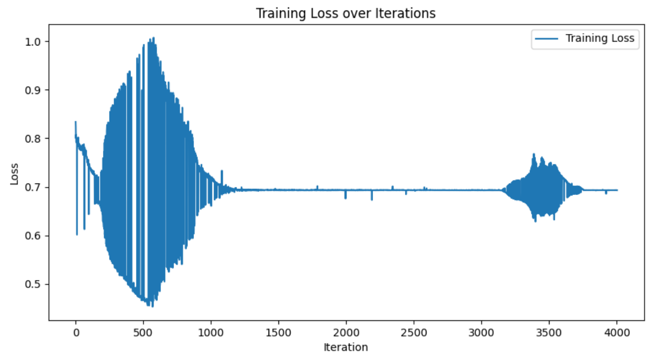
5.2. DeiT Finetuning

For our second experiment, we finetuned the DeiT model on our preprocessed images. We then used the Adam optimizer with a cross-entropy loss function to finetune it. We were able to perform 3 epochs of training with a batch size of 4 and a learning rate of 0.001.

6. Results

6.1. mini U-Net: Results

Despite not seeing the entire dataset, our model seemed to learn generally the correct labels. The loss fluctuated between 1-0.2 though generally stabilized around 0.693 over 1200 iterations. A plot of the loss is shown below.

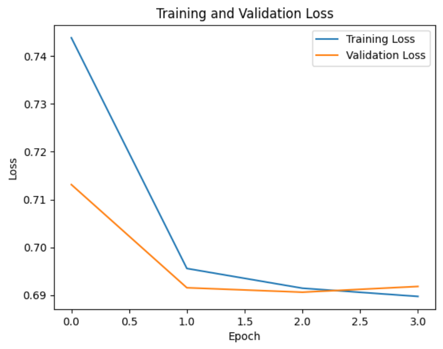


On the subset of 400 test examples, our model achieved the following metrics:

Accuracy	Precision	Recall	F1	AUROC
0.6509	0.4236	0.6509	0.5132	0.5

6.2. DeiT: Results

We found that finetuning the model was able to gradually decrease training and validation loss throughout training.



We ran into issues running our saved model on the test data so we were unable to find test results. However, we did have a validation accuracy of 54 percent.

7. Discussion

7.1. mini U-Net: Discussion

In our original proposal, we posited that using an architecture specialized for medical image processing on raw sinogram data could improve the already strong metrics presented by Hooper et al. In our project, we were not able to directly prove this. The U-Net architecture itself is very memory-intensive, having 23 convolutional layers. Further, the dataset we chose to use for this project did not help us stay within our hardware constraints. Our 16GB of GPU memory ran out quickly when attempting to run sinogram data through our mini U-Net, which still had over 5.5 million learnable parameters [4]. Each sinogram file was large, typically ∼150MB, making all operations with them quite slow. Additionally, due to time and computation restrictions, we were unable to come close to the amount of training done in the original Hooper et al paper, which trained their models for 50 epochs each, while we were unable to finish one due to constraints [1]. Still, we were able to create a model that was able to train on and, to some degree, learn from this expansive 1.3TB dataset, adapting the original U-Net architecture to one that may be suitable for binary classification.

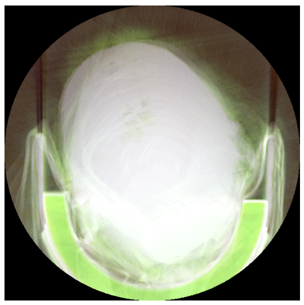
Each of our metrics are far below that achieved by Hooper et al, but by observing the provided loss graph, it is clear that the model did learn, but learned to guess the same label rather than learn features. This is likely due to the complexity of the features contained in the sinogram met with the reduced size of our mini U-Net and our limited training capabilities.

Our metrics confirm the idea that our model learned to guess the same label for each example, as the accuracy and recall are equivalent and the AUROC is 0.5, indicating that the test set was imbalanced, having 15.09% one label more than the other, and that the model did indeed guess the same label for each example at test time.

While these results for the mini U-Net are discouraging, they do not disprove our hypothesis, that a U-Net could outperform the Dense CNN when trained on raw sinogram data for HCT triage. Rather, these results show that this hypothesis requires a larger undertaking. With sufficient GPU and memory resources, it is still likely that a U-Net does outperform a Dense CNN on these tasks. However, with how much computation and memory was used even for evaluation, it begs the question of whether or not this additional compute is worth it for the accuracy boost.

7.2. DeiT: Discussion

Our experiment with finetuning DeiT on our own pre-processed images did not achieve the results we were hoping for. One hypothesis for this is that our pre-processing pipeline was flawed and distorted the original images. After already performing the pre-processing, we tried visualizing the images and found them to look like this:



Although we followed the standard procedure for CT windowing, we should have sanity checked each individual window to see that it worked before combining them. Additionally, I believe that training the model with more epochs could have improved the performance.

8. Conclusion & Future Work

We set out to find methods that would outperform a Dense CNN trained on raw sinogram data in HCT triage. One proposed method was training a U-Net, which is specialized for medical image segmentation, on raw sinogram data, while another was finetuning a DeiT transformer model on reconstructed images [4]. Ultimately we were able to achieve an accuracy of 0.54 with the DeiT model and 0.6509 for the U-Net, with some modifications to the originally proposed network. These figures were somewhat discouraging and can generally be attributed to tight memory and compute constraints.

Rather than disprove our original hypotheses, though, these figures show that this question is one that requires far more resources to answer. In future work, a team equipped with far more computational and graphical power could easily replicate and expand upon our work. However, before doing so, more work should be done to compare the costs of reconstructing images and using these HCT representations versus the computational cost of training a U-Net on large raw sinogram files. Further, our project does find that a mini U-Net architecture is able to learn, and perhaps could be more effective with smaller data. A simple and likely effective expansion on our work could be to use the mini U-Net architecture on the reconstructed HCT images and compare the findings to the original paper.

9. Acknowledgements

Jonathan set up our EC2 instance, implemented and trained the mini U-Net, evaluated the mini U-Net, and wrote all the corresponding materials in the paper.

Grant performed the formatting and downloading of the dataset onto our EC2 instance, implemented the vision transformer and finetuned DeiT, and wrote the corresponding materials in the paper. Our code can be found here

References

[1] S. M. Hooper, J. A. Dunnmon, M. P. Lungren, D. Mastrodicasa, D. L. Rubin, C. Ré, A. Wang, and B. N. Patel. Impact of upstream medical image processing on downstream performance of a head ct triage neural network. *Radiology: Artificial Intelligence*, 3(4):e200229, 2021.

[2] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[4] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[5] U. Ruby and V. Yendapalli. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[6] A. Shah, C.-Y. Wu, J. Mohan, V. Chidambaram, and P. Krähenbühl. Memory optimization for deep networks. *arXiv preprint arXiv:2010.14501*, 2020.

[7] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers distillation through attention, 2021.

[8] Y. Wu and K. He. Group normalization, 2018.

[9] T. Zhou, X. Ye, H. Lu, X. Zheng, S. Qiu, Y. Liu, et al. Dense convolutional network and its application in medical image analysis. *BioMed Research International*, 2022, 2022.