Elick Coval

5/4/2020

Computing for Health and Medicine

**Final Project – Defined**

**Project Description**

Our previous project performed a specific type of learning classed classification; the algorithm was trained on a test subset of images, dividing them into categories of skin lesions. This project has a very different task, it aims to use something called “survival analysis”, or time-to-event analysis. It utilizes images from the MNIST (Modified National Institute of Standards and Technology database), which is a database of handwritten digits, to predict a time to an event, using a specialized survival analysis loss function. The project goes on to generate synthetic survival data from these images and visualize it. Next, it uses the most widely used survival model for this type of problem, Cox’s Proportional Hazards Model, as a loss function to train a convolutional neural network using the LeNet architecture. Lastly, the convolutional neural network is then actually trained on the MNIST and predicts survival functions on the test data.

The project focuses on using digits from the MNIST as a substitute for patients in a clinical trial. Each digit is randomly assigned a risk score, each being unique, since no two real world patients are exactly the same. They are then grouped into one of 4 groups, 0 through 3, according to the risk score. For instance, “patient” 2 and 8 both have a risk score of around 0, so they are placed in group 0. These risk scores are then used to generate survival times which is then used to estimate the survival function. Real world patient data during a clinical trial can be precarious. Subjects can drop out early or the study can end before an event is captured, this is referred to as data censoring and is considered. A modified Cox’s model is then used to learn from the censored survival data. The end goal is realized when author then uses Breslow’s estimator to obtain estimated survival functions for images in the MNIST test data.

**Description of Data**

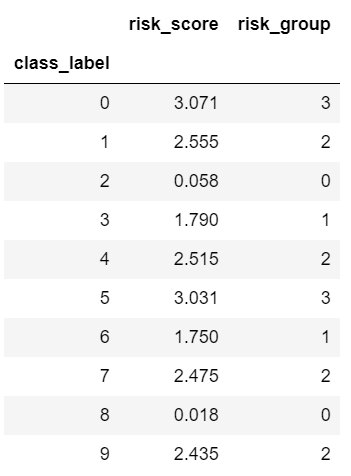
The data source for this project was a data set from the Modified National Institute of Standards and Technology database, or MNIST. It contains a dataset of 60,000 square grayscale images of handwritten single digits between 0 and 9. These digits were used to simulate different patients in a clinical trial, and is commonly used in testing machine learning algorithms, specifically convolutional neural networks because every digit is labelled, so it’s known beforehand what the digit is supposed to be. The digits 0 through 9 are assigned a risk score and grouped into risk groups which simulate patients who are in better or more dire straights in say a trial for a cancer drug.

**Related Work**

There are a couple of papers focused on the same area of study including Mobadersany et all (4) who uses the same method of using a CNN trained with the Cox model but instead of using the LeNet architecture they used the Visual Geometry Group architecture, “The SCNN combines elements of the 19-layer Visual Geometry Group (VGG) convolutional network architecture with a Cox proportional hazards model to predict time-to-event data from images.” (4) Another difference is that they are using actual microscopic images of brain tumors to predict overall survival of the patients, “Whole-slide images and clinical and genomic data were obtained from TCGA via the Genomic Data Commons (https://gdc.cancer.gov/). Images of diagnostic H&E-stained, formalin-fixed, paraffin-embedded sections from the Brain LGG and the GBM cohorts were reviewed to remove images containing tissue-processing artifacts, including bubbles, section folds, pen markings, and poor staining.” (4) While Mobadersany is using publicly available genomic and imaging data to predict cancer outcomes, Polsterl uses digit data from the MNIST database to simulate patients with different assigned “risk factors.”

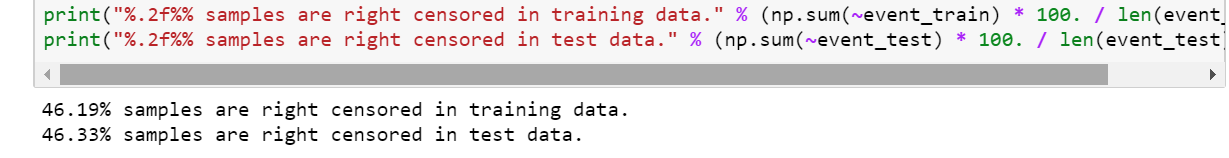
Another team using CNNs for survival analysis is Zhu et al (5) who also use Cox model, but their dataset is the National Lung Screening Trial lung cancer database. They, “develop a deep convolutional neural network for survival analysis (DeepConvSurv) with pathological images. The deep layers in our model could represent more abstract information compared with hand-crafted features from the images. Hence, it will improve the survival prediction performance.” (5) This is a similar approach to Mobadersany but focuses on lung cancer images instead of brained cancer images. They both, like Polsterl, use the Cox model to train a CNN to predict survival outcomes.

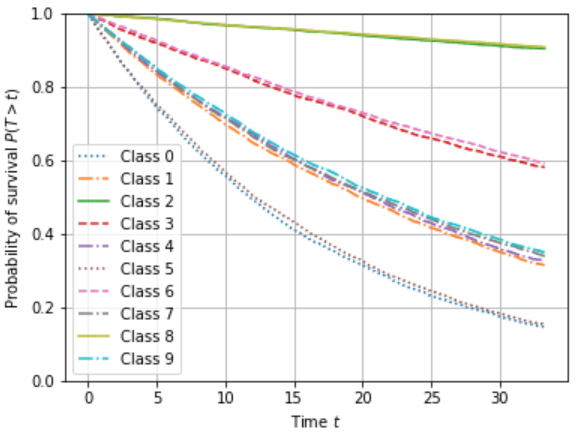
**Methods**

 To begin, the project used the images from the MNIST data set to generate survival times based on the digit each image represents, associating a risk score (survival time) with each of the ten digits in MNIST. It randomly assigns each class label to one of four overall risk groups, so some digits will simulate worse survival and others to better survival, simulating patients in a trial. Then, a risk score is generated to indicate how large the risk of an event happening is relative to the other numbers (patients).

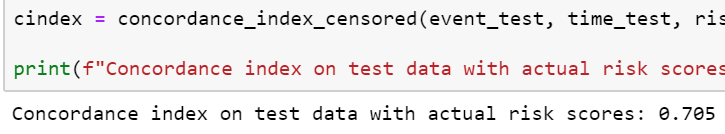
The snapshot to the left shows that class labels 0 and 5 belong to risk group 3, which is the highest risk score, while 2 and 8 belong to risk group 0, the lowest risk. Risk group 0 is close to 0 risk score, group 1 is around 1.7, while group 2 hovers around 2.5, and group 3 is close to 3. These risk scores are then used to generate survival times using the protocol of Bender et al (1) with the exponential distribution for survival time. It uses a probability desnity function of:

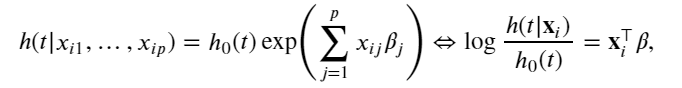
which is a scale parameter that is the inverse of the expectation: This results in a simple time-to-event model with no memory due to the constant hazard rate:

The author then chooses delta such that the mean survival time is 1 year and randomly censor survival times drawing times to approximately obtain 45% censoring. The generated data yields an observed time and a boolean event indicator for each MNIST image. This simulates a real clinical trial where patients would drop out of the trial and their outcome would be unknown, or the trial would end before an event occurred with some patients. It adds some real-world randomness/problems into the model. I obtained the same percentages as the project’s author.

The generated censored data is then used to estimate the survival function to see how the risk scores affect survival. This is accomplished using the non-parametric Kaplan-Meier estimator (2). I obtained a similar graph, as one would expect, and the classes with the lowest risk score have the highest probability of survival. Likewise, the classes with the highest risk score yield the lowest probability of survival.

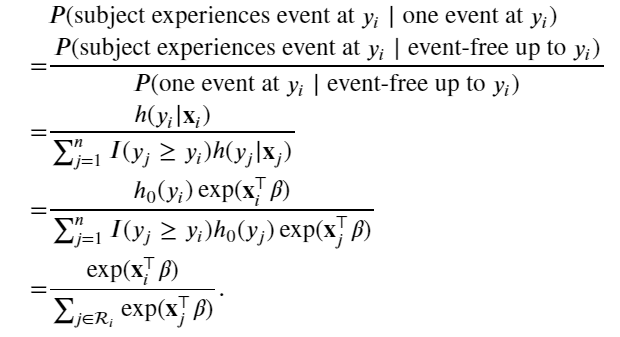
The project goes on to point out how important it is that both training and test data are subject to censoring because the exact time of an event, no matter how the data is split, cannot be observed. The author takes the risk score from which the survival time was generated and checks how good a model would perform if the actual risk score were known. I obtained a similar result.

 A perfect 1.0 result can never be obtained because the generated survival times are randomly distributed based on the risk scores and not deterministic functions of the risk score. Any model trained on this data will not exceed 0.705 for that reason.

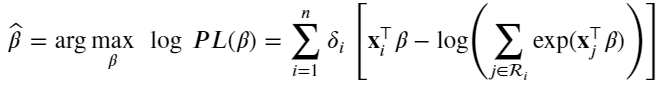
 Cox’s proportional hazards model (3) is the most widely used model to learn from censored survival data.

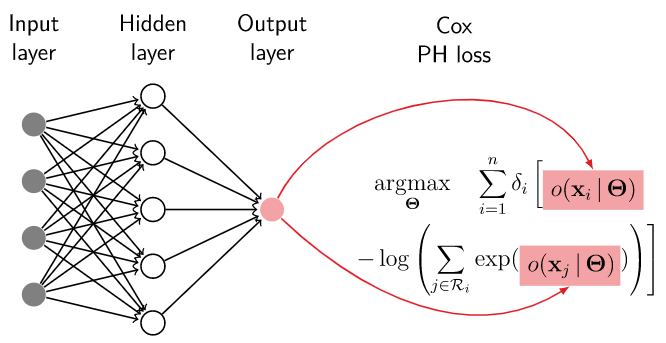
The hazard function is

split into two parts; the baseline hazard function h0 depends on time t, conversely the exponential is independent of time and only depends on the covariates xi. The model is fitted by maximizing the partial likelihood function.



By multiplying the conditional probablity for all patients who expereienced an event, and taking the logartihm, the partial likelihood function is found below.





This yields a linear model; however, a nonlinear model is needed for the neural network. The author solves this by replacing the linear predictor with the output of a neural network with parameters as shown on the right.

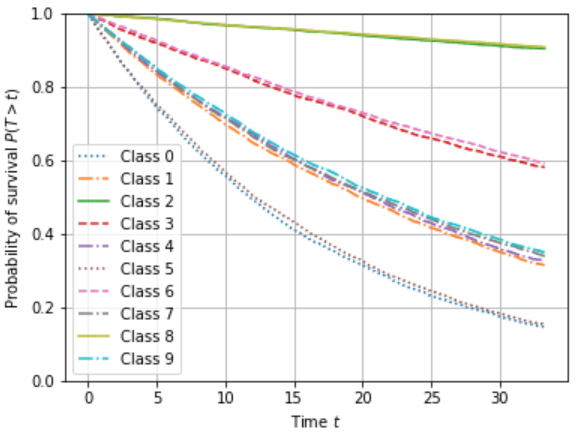
 The author runs into two problems when implementing the Cox loss function, one is the inner sum over the risk set: The other is that the risk set for the subject with the smallest uncensored survival time is over the whole dataset. He fixes the first problem by sorting the data once in descending order by survival time and then incrementally update the inner sum, which yields a linear complexity to compute the loss. The second problem is solved by using mini batches, as keeping the whole dataset in memory is unlikely. Exact loss cannot be computed because you may not have access to all samples since they would be split up, and you need to sort each mini batch by observed time individually, instead of the whole dataset at once. The author goes on to say this is usually fine as long as the batch contains several uncensored samples, otherwise the outer sum in the partial likelihood function would be over an empty set.

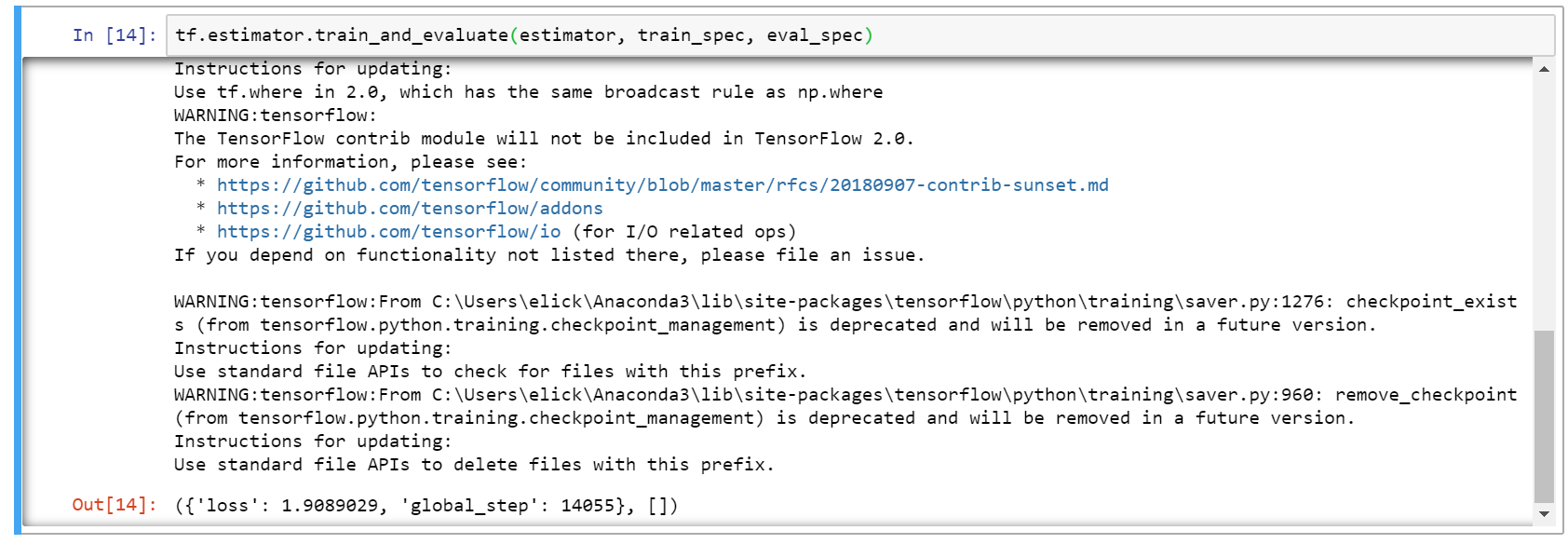
In addition to the Cox loss, the author computes the concordance index with respect to a separate validation set to monitor the training process. This index needs access to the entire dataset but, like was done with the Cox loss, cannot be done over a mini batch, as the estimated concordance index would be very volatile, which makes it hard to interpret. But the validation data is generally very much smaller than the training data, so collecting predictions for all the validation data and computing the concordance index is not a problem.

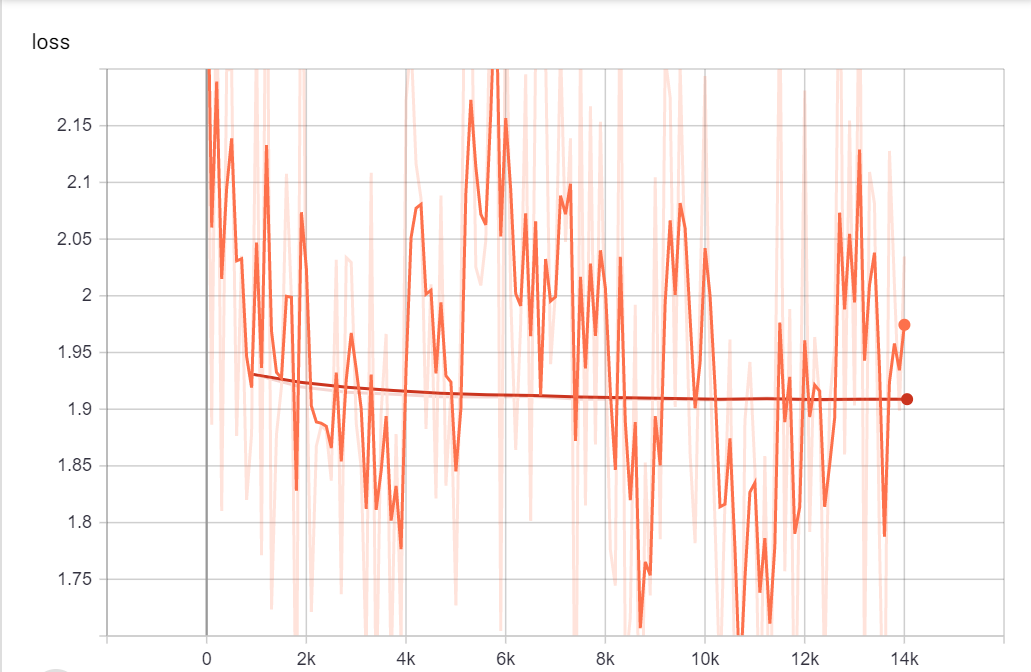
At last, it was finally time to create a convolutional neural network to learn a representation from MNIST digits so each image’s survival functions can be estimated. The CNN uses the LeNet architecture, the architecture typically used for the handwriting characters the project’s data set is comprised of. The last linear has one output unit that corresponds to the predicted risk score, this, together with the binary event indicator and risk set, are the inputs to the Cox loss function.

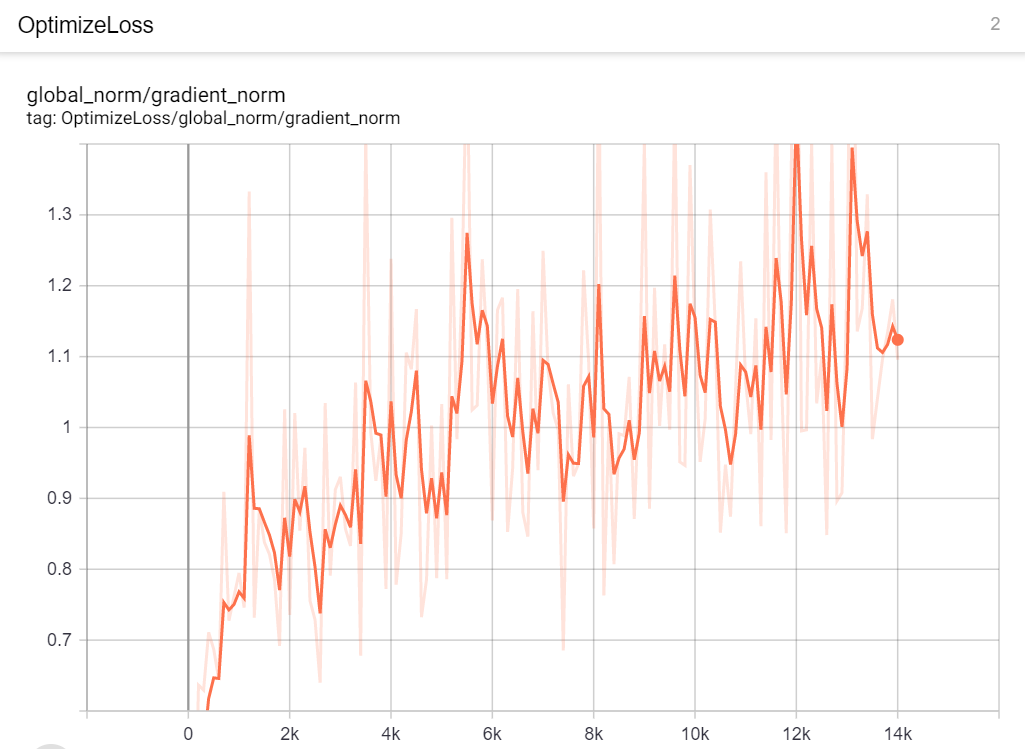
In order to estimate individual survival functions, a baseline hazard function needed to be estimated, which, as the author points out, can be done analogous to the linear Cox model using Breslow’s estimator (4). Once fitted, Breslow’s estimator can be used to estimate survival functions for images in the test dataset. Three samples are randomly drawn for each digit and their predicted survival functions are plotted. The resulting graph correctly displays the images that belong to risk group 0 (the lowest risk “patients”) has having the shallowest curve while the highest risk group having the deepest survival curve. This confirms that our model worked correctly and was able to learn which number corresponded to which risk group.

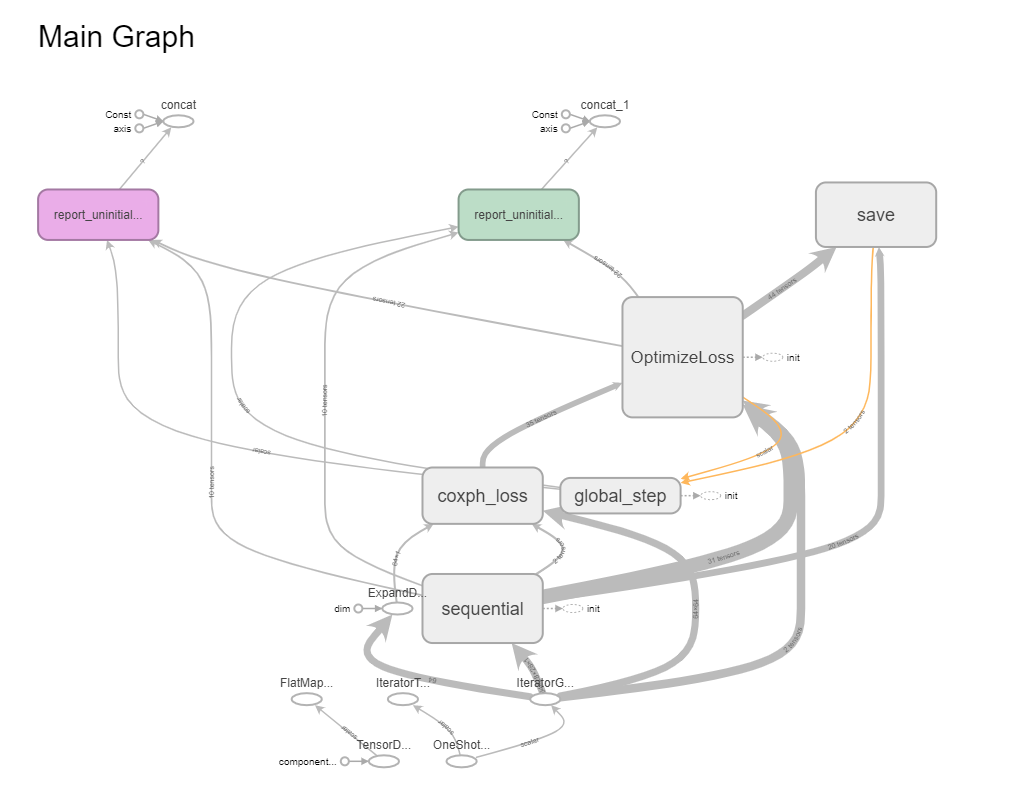
**Results**

 After using the generated censored data to estimate the survival function, the resulting graph corresponded with what was expected. The classes with the highest risk score, 0 and 5, had an increasingly lower probability of survival as time progressed. Likewise, the classes with the lower risk score, 2 and 8, had a relatively flat curve, indicating they had a higher chance of survival. This proves a good survival function was found.

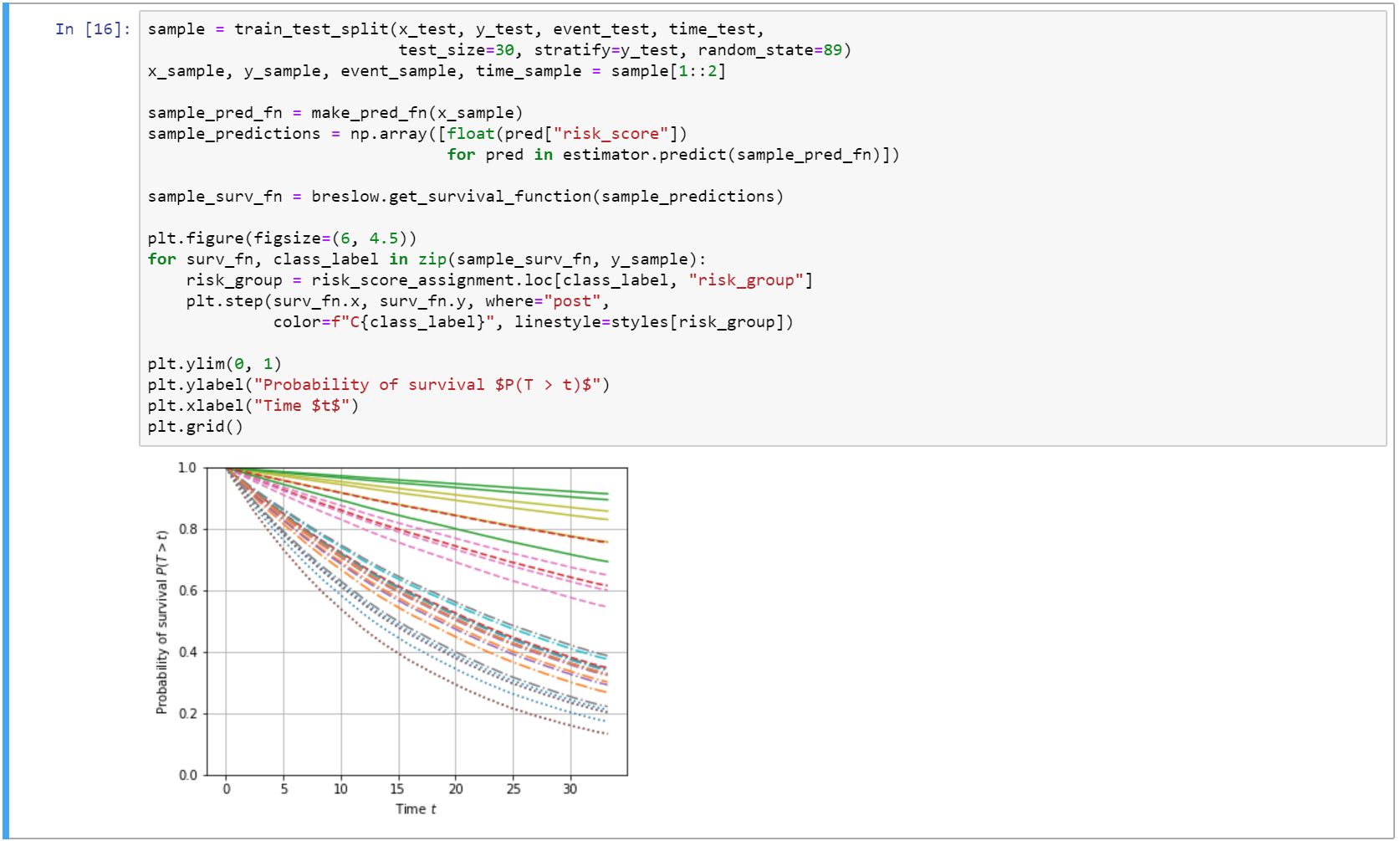
Although quite volatile, due to the small batch sizes (64) and the varying number of uncensored samples that contribute to the loss in each batch, the loss I obtained, 1.9089 was similar to the project’s author’s results.

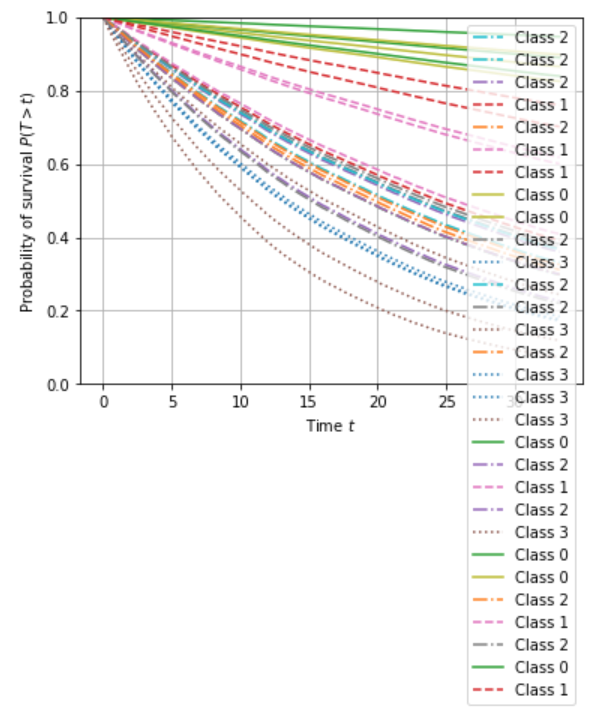
After running only 15 epochs it was good to see the loss heading towards 0, the lower the loss the better the model. The optimizer did a fantastic job of limiting the volatility and lowering the loss to around 1.1.

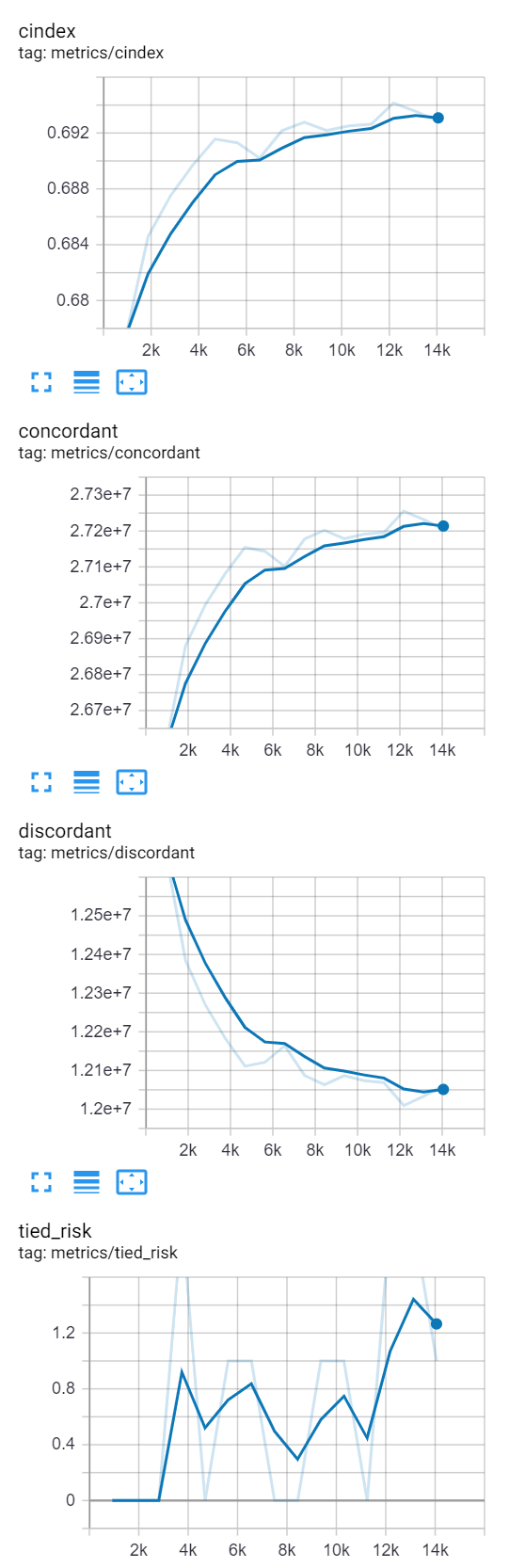


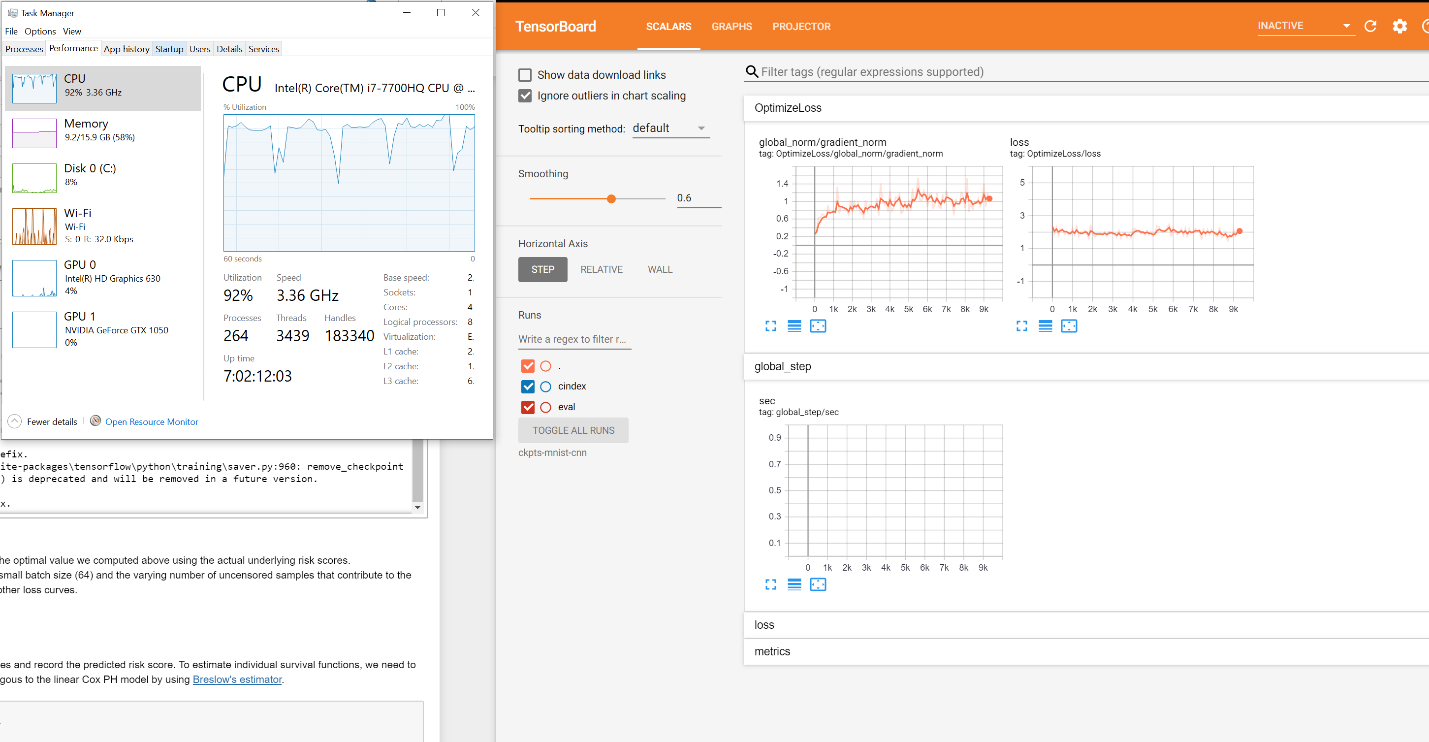


I found the image to the right helpful when trying to visualize what was happening with the CNN. There were many handy tricks that TensorBoard was able to perform. It is a great tool that I am going to dive more into in the future.

Finally, after randomly drawing three sample images for each digit and plotting their predicted survival function, they resulting graph looked like:

 Clearly the CNN did a fantastic job of predicting the survival probability from images given to it. The images belonging to the lowest risk groups had the flattest curve and the highest risk groups had the steepest curve. This is more easily seen when a legend is added to the graph.

 Some other results such as, concordance and c-index, are relevant in that, “the index of concordance is a "global" index for validating the predictive ability of a survival model. It is the fraction of pairs in your data, where the observation with the higher survival time has the higher probability of survival predicted by your model. As far as I remember it it equivalent to a rank correlation. The index is not calculated for every observation/subject. So the c-index can not be interpreted as the risk of a subject. High values mean that your model predicts higher probabilities of survival for higher observed survival times.” (6) The final c-index came very close the maximum the CNN could achieve which was found earlier as 0.705.

Below is just an image of the training session running on my CPU.

**Discussion**

My results were similar to the project’s author with a loss of ~1.9 then optimized to 1.1, which is fairly low and means the CNN did a great job at predicting. The cindex approached its theoretical maximum which was also good to see as this means the model was performing well. Also, the final graph showing probability of survival predictions of each class is further proof of how well this solution predicted outcomes for random digit “patients.”

**Future Work**

In the future it would be beneficial to run more epochs to determine if there would be any overfitting by the model. Also, this project uses a data set of random handwritten digits to simulate patients. It would be much more interesting if actual patient data were used similar to the projects referenced in the related work section. This is more difficult because it is hard to find prelabeled data which is needed for convolutional neural networks to learn. I am not going to pretend I know enough to advise changes to the loss function or some of the other more intricate details, but this is an extremely exciting area of study.

**Conclusion**

The task was to take a convolutional neural network and train it to predict time to a generated event from the given dataset, MNIST images. This was done using a modified Cox loss function specific to survival analysis. It is clear from the results, including the loss yielded, cindex and resulting graph that the CNN was able to accurately identify and predict a digit’s survival probability. As more covariates are identified they can be added to the model, but there are already researchers using methods like this to predict brain and lung cancer patient outcomes, which is very exciting. It was a lot of fun getting a taste of the sort of work being done at bleeding edge of this field of study.

**Citations**

1. https://scholar.google.com/scholar?cluster=11575471310627475868
2. https://en.wikipedia.org/wiki/Kaplan%E2%80%93Meier\_estimator
3. https://scholar.google.com/scholar?q=Cox%27s+Proportional+Hazards+Model&hl=en&as\_sdt=0&as\_vis=1&oi=scholart
4. https://www.pnas.org/content/115/13/E2970
5. https://ieeexplore.ieee.org/abstract/document/7822579
6. https://stats.stackexchange.com/questions/29815/how-to-interpret-the-output-for-calculating-concordance-index-c-index