Predicting Post-Procedural Complications Using MIMIC-III

Saeed Ul Hassan[1] and Faisal Maqbool[2]

***Abstract*— To predict outcomes and take decisions for patients admitted in ICU requires specific features of medical records: worth, capacity, access and dimensionality. A substantial amount of data needs to be stored in proper infrastructure and comprehensive analysis which can aid doctors, clinicians, medical experts and families is required. In this study, an investigation of medical data is carried out. We propose an ETL approach to extract features and train machine learning models to predict the post-procedural complications. To derive insights for that, we used well-known clinical dataset named Medical Information Mart for Intensive Care III (MIMIC-III) with two types of data sampling techniques: SMOTE and ADASYN. For both techniques our results showed that Random Forrest outperforms other linear models with an accuracy $>$ 80 %.**

I. INTRODUCTION

With advancements in digital technologies in medical sciences, different techniques making it possible to explore big data precisely and predict tasks of clinicians, labs and doctors using patient records of diagnoses or procedures. Diagnostic and medical technologies have evolved rapidly and doctors have to take complex decisions. Unfortunately, majority of clinical decisions are not based on evidence. According to 2012 Institute of Medical Committee Report, a small percentage of decision were evidence based. This problem ranges to not have proper clinical guidelines.......

In this paper, we implemented an ETL technique motivated from the different methodologies of ETL techniques [15], [16], which obtain data from original source to perform informative analysis and features extraction to extract features against not so rare diagnoses and procedures with defined ETL process, balancing of our dataset using distributions and sampling techniques and predict 996 (complication or no complication).

The International Classification of Diseases (ICD) [17] is the foundation which identifies and evaluates the statistics of diseases globally. Creates standards for all the diagnoses, diseases which further can be used for research purposes. Under revision of ICD9 codes, the code 996 defines complications particular to certain specified procedures and diagnoses. Most complications are caused due to cardiac, vascular or other used devices and some of them relates to reaction caused due to a procedure performed. In our work, we are focusing on such complications and investigating if those can lead to the mortality of patient. HIPAA (Health Insurance Portability and Accountability Act of 1996) [37] provides privacy and protection of health records for

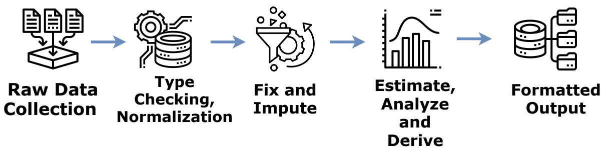

Fig. 1: Pre-processing steps applied on MIMIC-III .

safeguarding and protecting (PHI) medical information. To protect health information MIMIC-III provided anonymized data, still we need to make sure we are following HIPPA compliance rules so that our research does not conflict with any of the standards defined.

A predictor that can evaluate complications using the EHR can help hospital administrators, experts, families and doctors to take decisions that are required for the safety of the patient. For any hospital if the management already knows the precise expected stay of patient they can plan better to use their resources to provide the best possible care. For doctors predictors can evidence, statistics and labels which can be used for taking valuable decisions, for families who will pay for the expenses against the patient can better plan their billing using such predictors.

In our effort, we used Medical Information Mart for Intensive Care MIMIC-III [1]. MIMIC-III is a large, freely-available relational database comprising de-identified [38] health related data associated with over forty thousand patients who stayed in Intensive Care Units at Beth Israel Deaconess Medical Center (Boston, Massachusetts). The data spans June 2001 October2012. The ETL and predicting methods used demographics, data from different hospital systems, lab events, diagnoses, notes and other engineered information regarding each patient.

After the pre-processing step, we used two data sampling techniques: SMOTE and ADASYN and applied Logistic Regression, Random Forrest, Linear SVC and Neural Network to predict the post-procedural complications and accuracy results of all the models were recorded.

The paper is organized as follow: In section II, we explain the methods and frameworks used. In section III, we describe our experimental setup and execution of models. Further, we discussed the significance of results in section IV and concluded the paper in section V.

[1]Jeffrey is a PhD student in Materials and Analytical Sciences, University of Warwick, CV4 7AL Email: j.m.ede@warwick.ac.uk

[2]Richard is a Reader in the Deparment of Physics, University of Warwick, CV4 7AL Email: r.beanland@warwick.ac.uk

A. Literature Review

Numerous researches have also been conducted on the usage of MIMIC-III for creating new possibilities of research and scientific areas. As we are working with a huge medical data mart, the volume, diversity, dimensions and ETL matters. A. G. Rapsang and D. C. Shyam at el presented a study to identify the patient cohort from whole mart by searching the structured tables for diagnoses, procedures, chart events, demographics and other features is a burdensome task keeping the scoring systems [10] and V. H. Kury at el. authenticated the results of predictors with cost and time complexity in their prospective study as a computational retrospective study on MIMIC-II [11]. The studies with EHR databases that have both structural and unstructured data have not only help researchers in identifying the new possibilities but also helped them to recognize patients having higher risks [12].

Specific to MIMIC-III, from learning about the relationship between healthcare processes [35], the study [33] which made us realize that laboratory research is based on information retrieval irrespective of the design of study whether retrospective or prospective. All the management, doctors and clinicians are not involved in utilities of labs and how to proceed with MIMIC data and its structures. Another study [34] combined description of notes from the EHR and clinical trials using the state of the art NLP models to evaluate find relationships within trials and notes. Which, is our future plan to incorporate the textual features combining with complications to increase the performance of predicting complications accurately.

LOS prediction for the patient can provide valuable information to the management of the hospital but also for patient's health. [2] Explored the NN usage for predicting the LOS prediction within time range of (¡5) days or (¿5) days after the patient left the Intensive care unit. They used a subset of MIMIC-III and written all their models in R and PostgreSQL using a supercomputer provided by the Florida Polytechnic University. Their model predictions achieved 80 % accuracy and outperformed any other linear models previously used of predicting the length of stay. They used a neural net of 2 layers with 5 and 3 nodes at each level. In their final dataset, they had total 31018 data points, from which, they randomly chose subsets of 15000 points and trained the NN on the Florida Polytechnic University supercomputer.

Another study in 2019 [3] mapped ICD-9 codes using the clinical notes from physicians, doctors and other staff against each patient automatically using deep learning. Their main focus was to evaluate the performance of deep learning for mapping the ICD-9 codes. Their pipeline had extraction of data from Notes and Diagnoses, removing special characters and stop words from each note. They extracted non-sequential features and applied tfidf and word2vec. They baseline models LR, RF, FRNN on non-sequential features and on sequential features from word sequences + embedding matrix they applied Conv1D, LSTM and GRU. Overall they applied multiple experiments and showed that deep learning outperforms linear models to map ICD-9 codes with 0.6957 F1 score and accuracy of 0.8967.

Research published in BMC Medical Informatics and Decision Making [4] proposed a new approach by combining rule-based features and DL model with prior knowledge for effective disease classification. They evaluated their method on i2b2 obesity challenge and demonstrated that their model outperform other method used for disease classification. They based their method on Solt's Systems [43] (The Perl's Implementation of Solt provided by author) for first identifying the major triggers terms and phrases from notes, then predicting the classes with rare terms and phrases and then trained convolutional neural network. Another mortality prediction case study published in Machine Learning for HealthCare Conference [5] demonstrated large variability in studies of mortality prediction. They reviewed the performance of the related studies based on MIMIC-III and compared it to Gradient Boosting and LR ran on extracted data.

Apart from above mentioned researches a lot of other researchers contributed in predicting the complications. A study [6] explored the model performance in predicting after operation complications following anterior cervical discectomy and fusion (ACDF). They applied Logistic Regression, Random Forrest and ANN to achieve their results. 20,879 patients and ANN outperformed LR (p ¡ 0.05). Similarly, [7] used machine learning to derive and validate the hospital readmission. Their model reached (area under the receiver operating curve, 0.76). [8] A study with design of retrospective study, on predicting the real time complications used RNN against mortality, renal failure, and postoperative bleeding and operation revision. 47559 samples of ICU were included. Their deep learning models produced PPV 0·90 and sensitivity 0·85 for mortality, 0·87 and 0·94 for renal failure, and 0·84 and 0·74 for bleeding.

Another study using MIMIC-III [13] implemented a deep rule based fuzzy systems for predicting the accurate in-hospital mortality. Their main contribution to the system was proposing a system which can handle multiple categorical data.

II. Data

MIMIC-III (Medical Information Mart for Intensive Care III) database created by the Lab of Computational Physiology of The Massachusetts Institute for Technology (MIT) to fulfil the purpose of providing different techniques of data analysis and sciences. MIMIC-III is a large, freely-available relational database comprising de-identied [38] health related data associated with over forty thousand patients who stayed in Intensive Care Units at Beth Israel Deaconess Medical Center (Boston, Massachusetts). The data spans from June 2001 to October 2012.

The data collected from different ICU centers MICU, SICU,

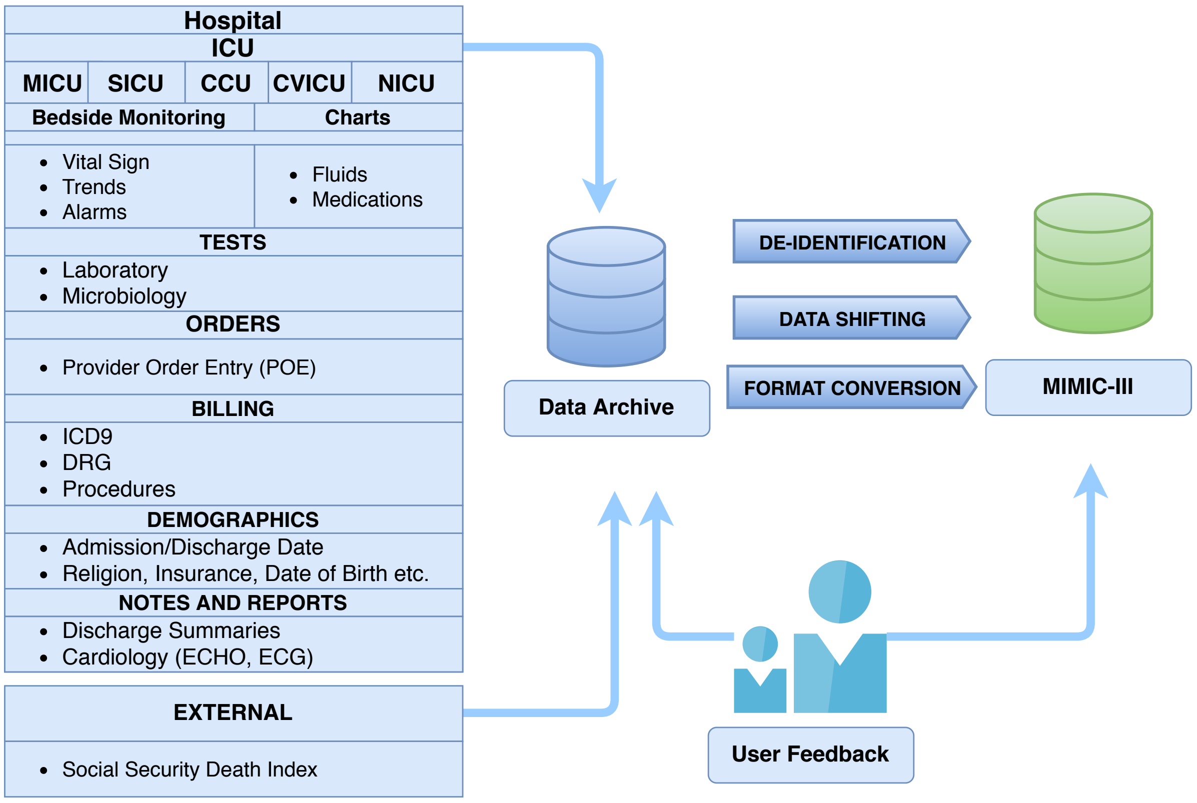

Fig. 2: MIMIC-III construction model

CCU, CVICU, NICU. The division of data includes bedside monitoring, charts, tests, orders, billing, demographics and notes. The archived data then passes through the steps of transformation which include de-identification, data shifting and format conversion along with user feedback to form the MIMIC-III.

MIMIC-III is a comprehensive collection of de-identified data from 53423 distinct critical care hospital admissions from 38597 distinct adult patients. The data has been compiled into 26 tables which contain, for example, an average of 4579 noted values and 380 lab charted measurements for each hospital admission as well as a total of 3.8 gigabytes of unstructured textual data from various healthcare provider notes and analyses. In addition to de-identifying patient data, MIT requires training in the protection of patient data for anyone requesting access to the MIMIC database.

The active researchers have contributed to already given data with additional scripts to generate new concepts and insights at MIMIC code repository which includes views and tables as well.

To complete ETL process, we used PostgreSQL and Python. Multiple SQL scripts were written for creation of tables, indexes, materialized views and derived tables. All of which are presented on a public repository [19] 4. Extraction of major chart events and lab events against each patient involved filtering of specific patients, lookup against particular diagnoses and procedures which we did under the batch processing of huge event files using Python. Following are the steps used:

Item	Count
Diagnoses	14328
Procedures	3882

TABLE I: Procedures and Diagnoses Count

1) Reading files to Dataframe.
2) Subset each batch.
3) Filter features from each batch.
4) Lookup for chart events, diagnoses and procedures
5) Compile the features set

Item	Count
Complication	2754
No Complication	30491

TABLE II: Class Count

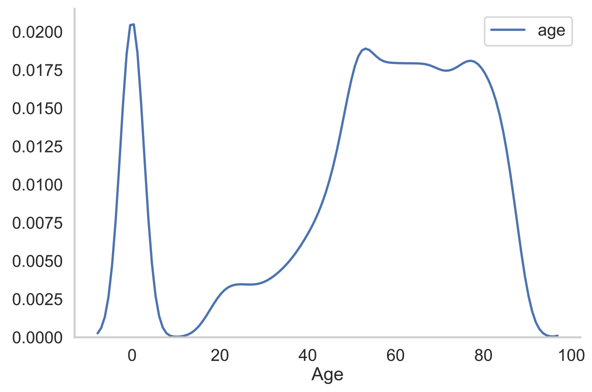

Fig. 3: Convolutional image 2× supersampling network with three skip-2 residual blocks.

A. *Feature Space*

Potentials features which we selected from feature space are as follow:

- **General:** Insurance, Martial status, Hospital Expire Flag, Length of Stay, Calculated Bicarbonate,TotalCo2, Chloride, Free Calcium, Glucose, Hematocrit, Hemoglobin, Lactate, Oxygen, Oxygen Saturation, PCO2, PH, Potassium, Sodium, Temperature, Calcium Total, Centromere, Creatinine, Globulin, Blood Glucose, Blood Lipase, Blood Magnesium, Blood Potassium, Blood Sodium, Platelets Counts, Red Blood Cells, White Blood Cells, Lymphocytes
- **Engineered Concepts:** Hypothyroidism, Renal Failure, Liver Disease, Peptic Ulcer, Aids, Lymphoma, Metastatic Cancer, Solid Tumor, Rheumatoid Arthritis, Coagulopathy, Obesity, Weight Loss, Fluid Electrolyte, Blood Loss Anemia, Alcohol Abuse, Drug Abuse, Psychoses, Depression, Congestive Heart Failure, Cardiac Arrhythmias, Valvular Disease, Pulmonary Circulation, Peripheral Vascular, Hypertension, Paralysis, Other Neurological, Chronic Pulmonary, Diabetes Uncomplicated, Diabetes Complicated

III. Experimental Setup

As we are working with binary classification in which either a patient will have complications or not. We used accuracy as the metric of performance to validate our models which is defined as:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\mathrm{i}_i{=}\mathrm{i}_i^{'}) \qquad (1)$$

A. *Oversampling*

To overcome the imbalances in our classes of complications vs non-complications, we performed two oversampling techniques.

1) SMOTE: SMOTE [33] proposed by Chawla et al., is an oversampling method. This method interpolate the minority class neighbors to construct new minority class samples randomly. The method can be described as follows. First, for each minority class sample x one gets it's k nearest neighbors from other minority class samples. Second, one chooses one minority class x' sample among k neighbors. Finally, one generates the synthetic sample x(new) by interpolating between x and x' as follows:

$$X_{new} = x + rand(0,1) * (x^{'} - x) \qquad (2)$$

Where (0, 1) refers to random number between (0,1]. The data gets interpolated between minority class samples by expanding the minority class samples. By using SMOTE we can also avoid the over-fitting problem.

2) ADASYN: ADASYN [34] is another oversampling method which interpolates new minority class samples by first calculating the number of synthetic samples of minority class and for each minority sample find the nearest neighbors by calculating the Euclidean distance.

The major difference between SMOTE and ADASYN is the difference in the generation of synthetic sample points for minority data points. In ADASYN, we consider a density distribution r which thereby decides the number of synthetic samples to be generated for a particular point, whereas in SMOTE, there is a uniform weight for all minority points.

B. *Models*

1) Linear SVC: A linear SVC support vector classifier (SVM) a supervised predictor which given a labeled dataset finds the ”best fit” hyperplane which divides and segregate the data into different classes. It has the cost function like logistic regression defined below:

$$P_{\Theta}(x) = \begin{Bmatrix} 1 & if\Theta^{T}x >= 0 \\ 0 & otherwise \end{Bmatrix} \qquad (3)$$

2) Random Forrests: Random forests [**?**] are predictors which on given labeled data create random forests for reaching towards the target of each instances and map all the features to those trees. The actual output is then generated by averaging the output of all the forests. Tin Kam Ho initially presented the concept of Random Forrests in the year of 1995. In 2001 [**?**] where Breimen extend the concept by extending already built algorithm and work of Amit [**?**]. The random forest creates many decision trees from which it calculates the actual and optimal path to find the expected label. It average the performance from all the forests, the major concepts due to which model was named Random Forrest are as follow:

- While building new forests, sample the data in random fashion
- While splitting the parent and leaf nodes, creation of subsets of multiple features

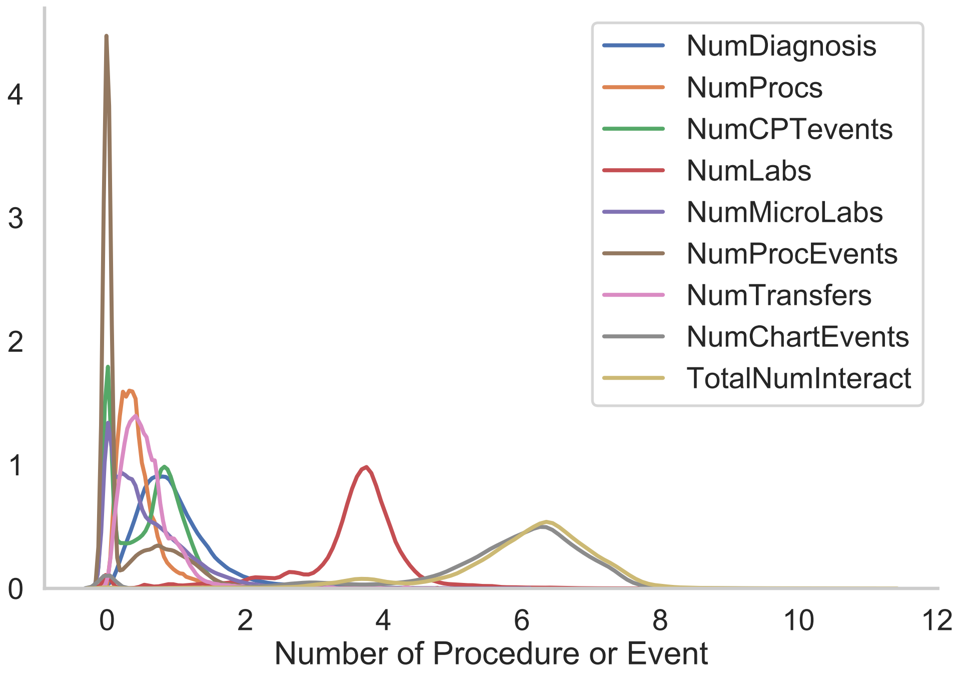

Fig. 4: Distribution of Different Events

3) Boosting: boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.

The boosting ensemble technique consists of three simple steps:

- An initial model F0 is defined to predict the target variable y. This model will be associated with a residual $(y – F_0)$
- A new model h_1 is fit to the residuals from the previous step
- Now, F_0 and h_1 are combined to give F_1, the boosted version of F_0. The mean squared error from F_1 will be lower than that from F_0:

$$F_1(x) \leftarrow F_0(x) + h_1(x) \tag{4}$$

To improve the performance of F1, we could model after the residuals of F1 and create a new model F2:

$$F_2(x) \leftarrow F_1(x) + h_2(x) \tag{5}$$

This can be done for ‘m’ iterations, until residuals have been minimized as much as possible:

$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x) \tag{6}$$

Results: Example learning curves for mean squared and quartic error training are shown in fig. 6. Training is more stable and converges to lower losses for larger batch sizes.

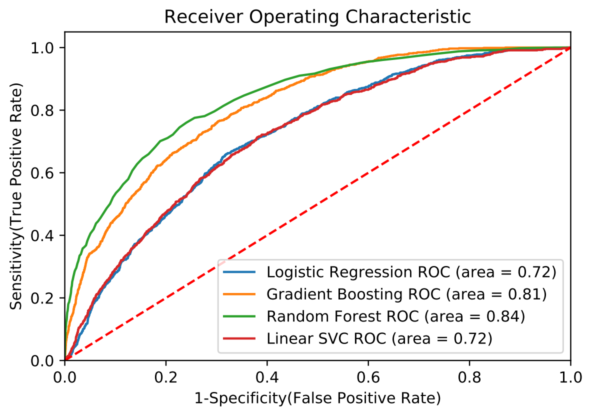

Fig. 5: Convolutional image $2\times$ supersampling network with three skip-2 residual blocks.

Training is less stable for quartic errors than squared errors, allowing ALRC to be examined for loss functions with different stability.

Training was repeated 10 times for each combination of ALRC threshold and batch size. Means and standard deviations of the means of the last 5000 training losses for each experiment are tabulated in table III. ALRC has no effect on mean squared error (MSE) training, even for batch size 1. However, it decreases errors for batch sizes 1, 4 and 16 for mean quartic error training.

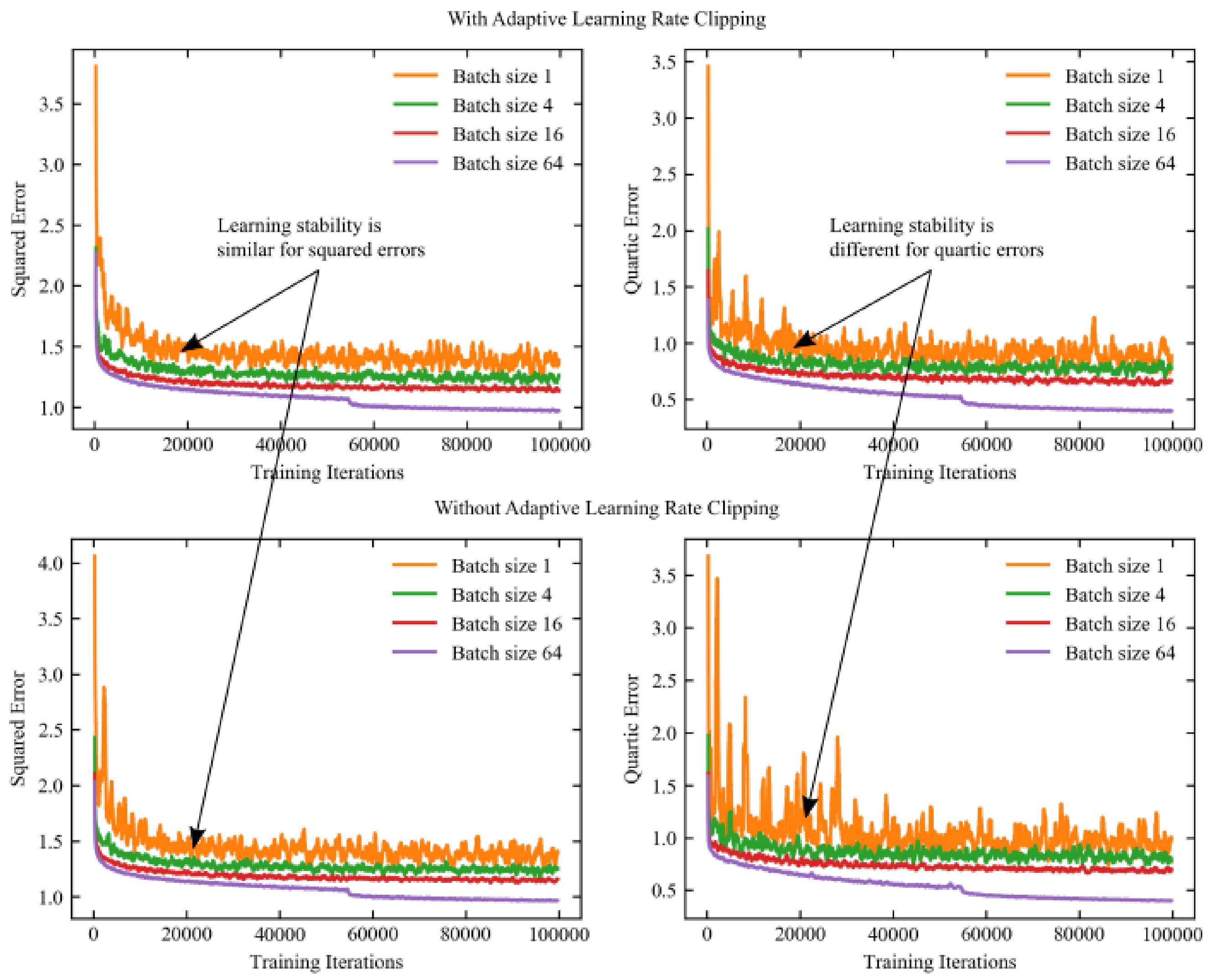

Fig. 6: Unclipped learning curves for 2× CIFAR-10 upsampling with batch sizes 1, 4, 16 and 64 with and without adaptive learning rate clipping of losses to 3 standard deviations above their running means. Training is more stable for squared errors than quartic errors. Learning curves are 500 iteration boxcar averaged.

Count Statistics

	Items		Classes		Expiry vs Complications		Batch Size 64	
Threshold	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
2	5.55	0.048	4.96	0.016	4.58	0.010	-	-
3	5.52	0.054	4.96	0.029	4.58	0.004	3.90	0.013
4	5.56	0.048	4.97	0.017	4.58	0.007	3.89	0.016
∞	5.55	0.041	4.98	0.017	4.59	0.006	3.89	0.014

Quartic Errors

	Batch Size 1		Batch Size 4		Batch Size 16		Batch Size 64	
Threshold	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
2	3.54	0.084	3.02	0.023	2.60	0.012	1.65	0.011
3	3.59	0.055	3.08	0.024	2.61	0.014	1.58	0.016
4	3.61	0.054	3.13	0.023	2.64	0.016	1.57	0.016
∞	3.88	0.108	3.32	0.037	2.74	0.020	1.61	0.008

TABLE III: Adaptive learning rate clipping (ALRC) for losses 2, 3, 4 and ∞ running standard deviations above their running means for batch sizes 1, 4, 16 and 64. ARLC was not applied for clipping at ∞. Each squared and quartic error mean and standard deviation is for the means of the final 5000 training errors of 10 experiments. ALRC lowers errors for unstable quartic error training at low batch sizes and otherwise has little effect. Means and standard deviations are multiplied by 100.

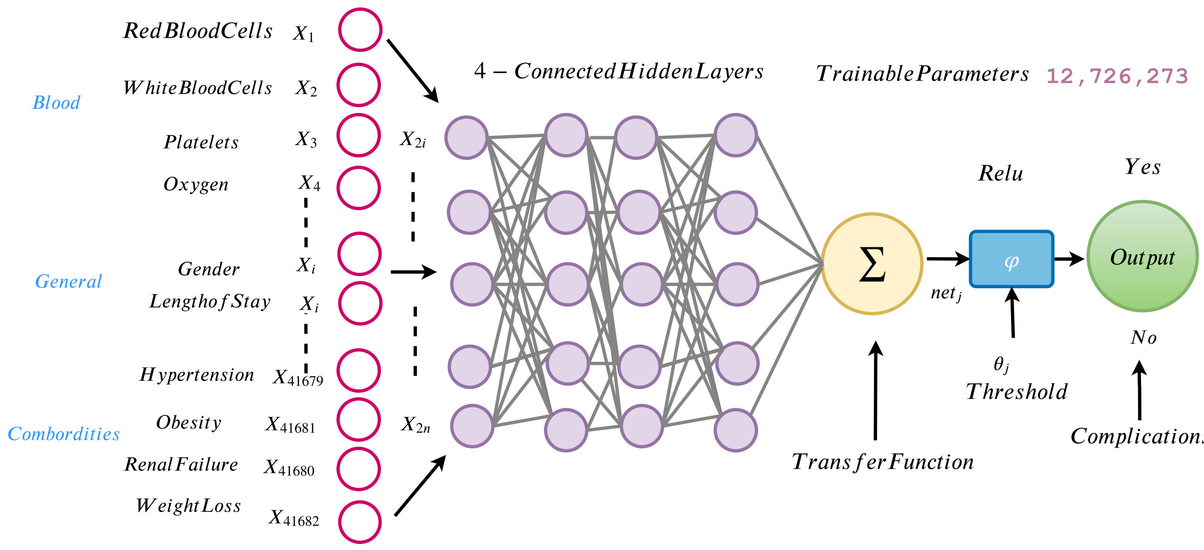

Fig. 7: Neural network completions of 512×512 scanning transmission electron microscopy images from 1/20 coverage blurred spiral scans.

IV. Experiments: Partial-STEM

To test ALRC in practice, we applied our algorithm to neural networks learning to complete 512×512 scanning transmission electron microscopy (STEM) images from partial scans with 1/20 coverage. Example completions are shown in fig. 7.

Data pipeline: In order, each image was subject to a random combination of flips and 90° rotations to augment the dataset by a factor of 8. Next, each STEM images was blurred and a path described by a 1/20 coverage spiral was selected. Finally, artificial noise was added to scans to make them more difficult to complete.

Architecture: Our network can be divided into the three subnetworks shown in fig. 8: an inner generator, outer generator and an auxiliary inner generator trainer. The auxiliary trainer[1, 2] is introduced to provide a more direct path for gradients to backpropagate to the inner generator. Each convolutional layer is followed by ReLU activation, except the last.

Initialization: Weights were initialized from a normal distribution with mean 0.00 and standard deviation 0.05. There are no biases.

Weight normalization: All generator weights are weight normalized[3] and a weight normalization initialization pass was performed after weight initialization. Following [3, 4], running mean-only batch normalization was applied to the output channels of every convolutional layer except the last. Channel means were tracked by exponential moving averages with decay rates of 0.99. Similar to [5], running mean-only batch normalization was frozen in the second half of training to improve stability.

Loss functions: The auxiliary inner generator trainer learns to generate half-size completions that minimize MSEs from half-size blurred ground truth STEM images. Meanwhile, the outer generator learns to produce full-size completions that minimize MSEs from blurred STEM images. All MSEs were multipled by 200. The inner generator cooperates with the auxiliary inner generator trainer and outer generator.
To benchmark ALRC, we investigated training with MSEs, Huberized ($h = 1$) MSEs, MSEs with ALRC and Huberized ($h = 1$) MSEs with ALRC before Huberization. Training with both ALRC and Hubarization showcases the ability of ALRC to complement another loss function modification.

Learning policy: ADAM optimization[6] was used with a constant generator learning rate of 0.0003 and a first moment of the momentum decay rate, $\beta_1 = 0.9$, for 250000 iterations. In the next 250000 iterations, the learning rate and β_1 were linearly decayed in eight steps to zero and 0.5, respectively. The learning rate for the auxiliary inner generator trainer was two times the generator learning rate; β_1 were the same. All training was performed with batch size 1 due to the large model size needed to complete 512×512 scans.

Results: Outer generator losses in fig. 9 show that ALRC and Huberization stabilize learning. Further, ALRC accelerates MSE and Huberized MSE convergence to lower losses. To be clear, learning policy was optimized for MSE training so direct loss comparison is uncharitable to ALRC.

V. Discussion

Taken together, our CIFAR-10 supersampling results show that ALRC improves stability and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Loss spikes are often encountered when training with high learning rates, high order loss functions or small batch sizes. However, a moderate learning rate was used in MSE experiments so losses did not spike enough to destabilize learning. In contrast, mean quartic error training is unstable so ALRC stabilizes training and lowers losses. Similar results are confirmed for partial-STEM where ALRC

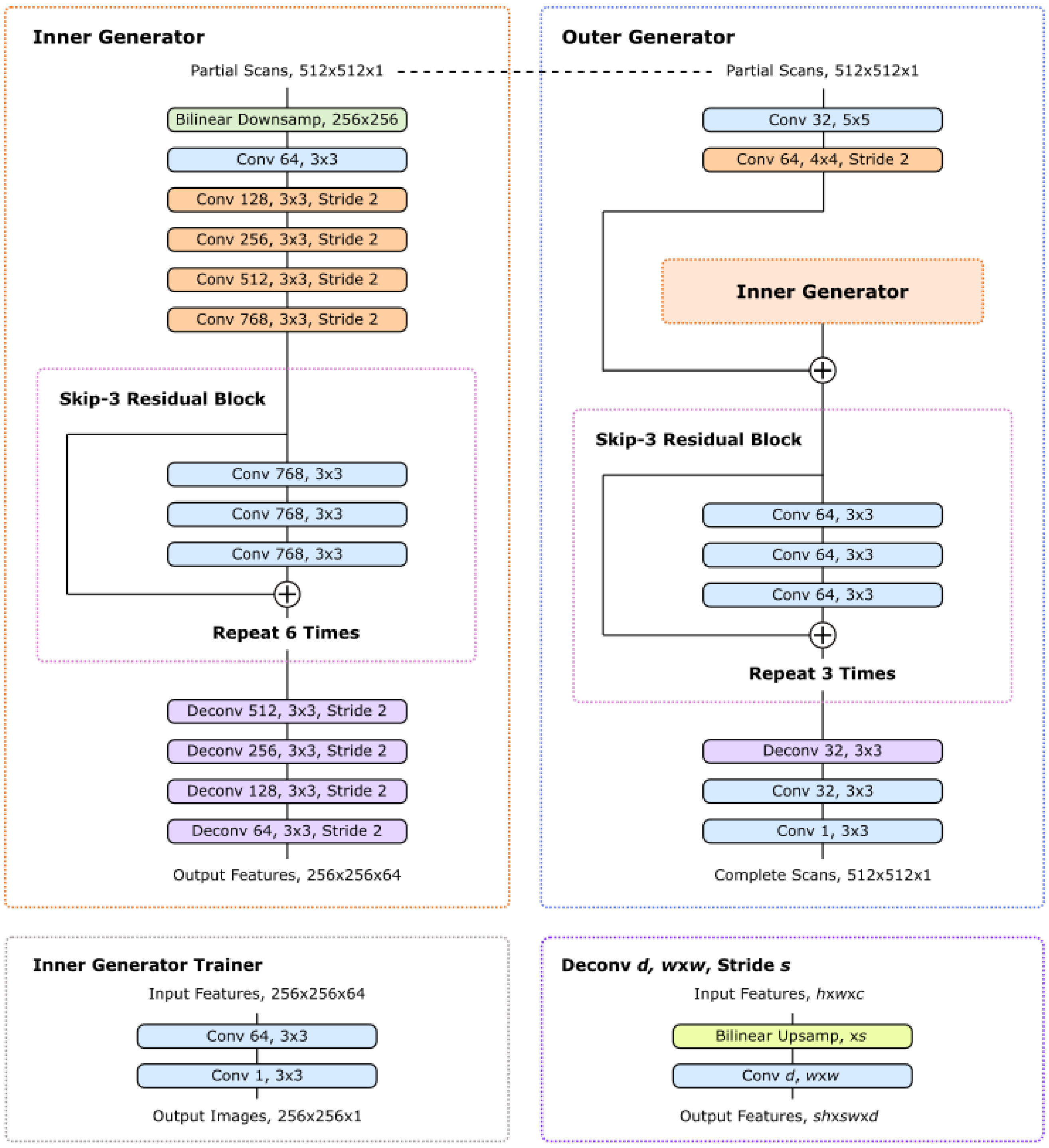

Fig. 8: Two-stage generator that completes 512×512 micrographs from partial scans. A dashed line indicates that the same image is input to the inner and outer generator. Large scale features developed by the inner generator are locally enhanced by the outer generator and turned into images. An auxiliary inner generator trainer restores images from inner generator features to provide direct feedback.

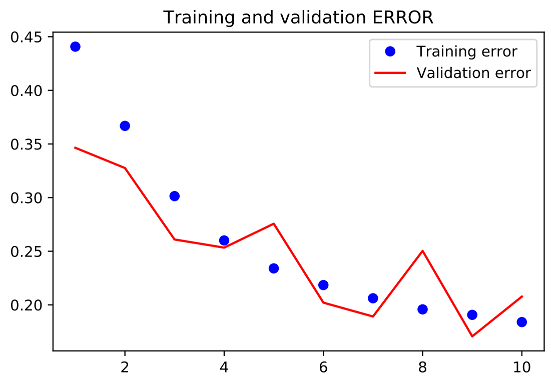

Fig. 9: Outer generator losses show that ALRC and Huberization stabilize learning. ALRC lowers final mean squared error (MSE) and Huberized MSE losses and accelerates convergence. Learning curves are 2500 iteration boxcar averaged.

stabilizes learning and lowers losses.

ALRC is designed to complement existing learning algorithms with new functionality. It is effective for any loss function or batch size and can be applied to any neural network trained with a variant of stochastic gradient descent. Our algorithm is also computationally inexpensive, requiring orders of magnitude fewer operations than other layers typically used in neural networks. As ALRC either stabilizes learning or has little effect, this means that it is suitable for routine application to arbitrary neural network training with SGD. In addition, we note that ALRC is a simple algorithm that has a clear effect on learning.

Nevertheless, ALRC can replace other learning algorithms in some situations. For instance, ALRC is a computationally inexpensive alternative to gradient clipping in high batch size training where gradient clipping is being used to limit perturbations by loss spikes. However, it is not a direct replacement as ALRC preserves the distribution of backpropagated gradients whereas gradient clipping reduces large gradients. Instead, ALRC is designed to complement gradient clipping by limiting perturbations by large losses while gradient clipping modifies gradient distributions.

The implementation of ALRC in algorithm **??** is for positive losses. This avoids the need to introduce small constants to prevent divide-by-zero errors. Nevertheless, ALRC can support negative losses by using standard methods to prevent divide by zero errors. Alternatively, a constant can be added to losses to make them positive without affecting learning.

ALRC can also be extended to limit losses more than a number of standard deviations below their mean. This had no effect in our experiments. However, preemptively reducing loss spikes by clipping rewards between user-provided upper and lower bounds can improve reinforcement learning[7]. Subsequently, we suggest that clipping losses below their means did not improve learning because losses mainly spiked above their means; not below. Some partial-STEM losses did spike below; however, they were mainly for blank or otherwise trivial completions.

VI. Conclusions

We have developed ALRC to stabilize the training of artificial neural networks by limiting backpropagated losses. Our experiments show that ALRC accelerates convergence and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Further, ALRC is computationally inexpensive, can be applied to any loss function or batch size, does not affect the distribution of backpropagated gradients and has a clear effect on learning. Overall, ALRC complements existing learning algorithms and can be routinely applied to arbitrary neural network training with SGD.

VII. Source Code

Source code for CIFAR-10 supersampling experiments and a TensorFlow[8] implementation of ALRC is available at https://github.com/faisalmaqbool94/Predicting-Post-Procedural-Complications-Using-MIM

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions, corr abs/1409.4842," *URL http://arxiv.org/abs/1409.4842*, 2014.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision. arxiv 2015," *arXiv preprint arXiv:1512.00567*, vol. 1512, 2015.

[3] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

[4] E. Hoffer, R. Banner, I. Golan, and D. Soudry, "Norm matters: efficient and accurate normalization schemes in deep networks," in *Advances in Neural Information Processing Systems*, pp. 2160–2170, 2018.

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

VIII. Acknowledgements

This research was funded by EPSRC grant EP/N035437/1.