Auxiliary feature learning for small dataset regularization (Supervised learning)

Madhavan Krishnan[1]
[1]School of Computer Science and Electronic Engineering,
University of Essex, Colchester, UK

***Abstract*— In this paper, a novel technique is presented which allows the deep learning model to learn auxiliary features from a small set of data. These features are important for the model's ability to generalize and perform well on unseen example without the need for huge data-set.**

I. INTRODUCTION

Deep neural networks presence seems everywhere, mainly in the field of computer vision, speech recognition, and reinforcement learning. This is due to its magical ability to perform extraordinarily well in pattern recognition tasks. In spite of the state of the art performance with various deep learning architecture, more concerns are raised on the lack of transparency of these model[9].

Due to this black box model, it is some times hard to justify the actions of the model or even understand the reasoning behind the model's decision. The main concern is that the model may not necessarily be learning the right features to make a prediction. which will make the model perform poorly on the unseen data[5]. This can be attributed to the fact that the model has memorized the training data-set. This behavior is not desirable.

Most of the techniques like cross-validation and K-fold are devised to address this issue. To think of this in an intuitive sense, the model can be considered as a student that is preparing for an exam[16]. In an ideal case, the student learns the underlying concept and perform well on the test based on their understanding. but what more often than not happens is the student memorizes the entire material and during the test will fail to perform well.

In practice, as the number of layers increases in deep learning architecture, its ability to memorize the training example also increases. This nature of DNN is referred to as the hunger for more data-sets[29]. Which is contradicting to how humans learn, through small examples and generalization to new unseen ones with little to no effort[**?**]. This ”memorizing” behavior is being addressed in this approach using an auxiliary feature learning technique. The effect of this technique is addressed in the below sections.

Solving this problem will take us one step closer to solving generally intelligence. Not to mention the benefits for the real world applications in the field of health care, industries, and automation.

II. BACKGROUND

Most of the recent development in the field of deep learning is on the architecture of stacking up the layers. From VGG16 to Inception V4 to Resent 50 is on the premises of network architecture[32].

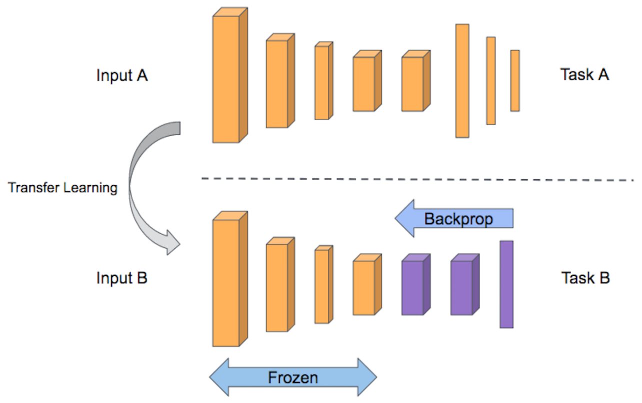


Fig. 1. Transfer learning architecture

Most of the generalization of these models (Fig 1) is attributed to the initial layers that were trained on Image-Net data-set for weeks in huge clusters of GPUs[35]. Image-Net consists of millions of images manually labeled across thousands of categories which are publicly available as open sources. Once that is trained. The weights of the models are frozen [37]-[41](especially on the initial layers)

It is now easy to do transfer learning on the final layers on these models to a similar problem and achieve decent results. with all the heavy lifting done in the earliest layers.

A most common approach when dealing with a small set of data during training is to have the data-set augmented. This is to increase the size of the data-set and hoping the model learn to generalize well and make correct predictions after training on the augmentation[1][3].

Few techniques on data augmentation by a random selection of sample pairs and interpolation between them and also their associated one hot encoded data is shown significant improvements in the model's ability to generalize[2]-[10].

Auxiliary tasks is another approach (Fig 2) that has shown good performance gain in the model ability to perform on the test and validation data-set. A typical method of training is to have a single task for which the model tries to learn the features and minimizes the objective function. This will lead

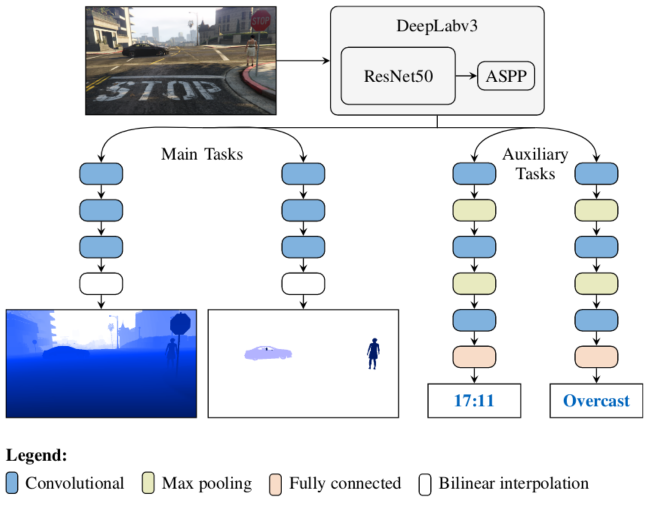


Fig. 2. Road scene analysis enhanced using auxiliary tasks

to over-fitting after a certain number of epochs[16]-[21]. If the model is tasked with not just one main task but, other supplementary tasks along with the main task, will force the model to learn more general features that can minimize the error function and not over-fit. For example a data-set on real-time segmentation of roads tasked with predicting the weather of the seen as shown statistically significant improvement in the result[11].

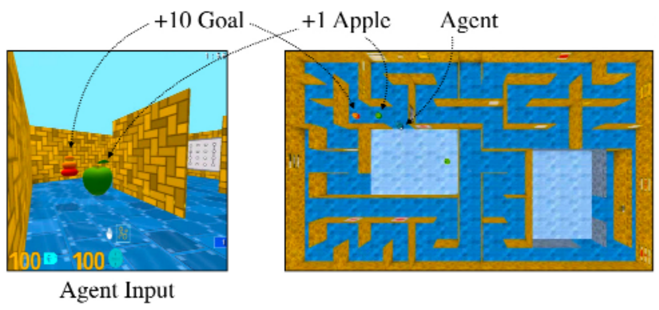


Fig. 3. Auxiliary reward function to guild the agent

Same goes of the field of reinforcement learning in Fig 3. Having some pseudo rewards that help the agent focus on the actual task will help guide the model's ability to learn quickly and converge to global minimum sooner[25].

III. METHODOLOGY

The general idea behind the auxiliary feature learning is to find the "best" features that help the models ability to learn and generalize on the unseen data-set.

Meaning passing noisy data as input the model should be able to isolate the noise and learn meaningful features to help tackle the task in hand[35]. Once these features are extracted, any set of machine learning techniques such as neural network, kernel-SVM, a Decision tree can be used to further find patterns in the features.

Auto-encoder is used in many applications from image denoising to image-compression. Since it is an unsupervised learning approach the need for a labeled data-set is not required. This technique is used in learning the auxiliary features by encoding the input data in a latent space and decoding it back from the latent space[8]. What it does is retrains the useful information to represent the original input data, by doing it will learn to remove the noise.

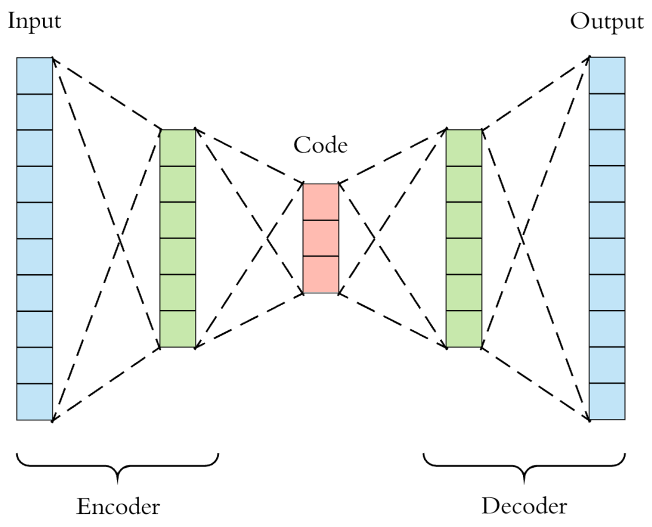


Fig. 4. Architecture of Auto-encoder

The experiment is set is such a way that, a small set of sample is taken from the data-set and used to train the Auto-encoder. This is in hopes to test the generalization of the approach on the entire data-sets.

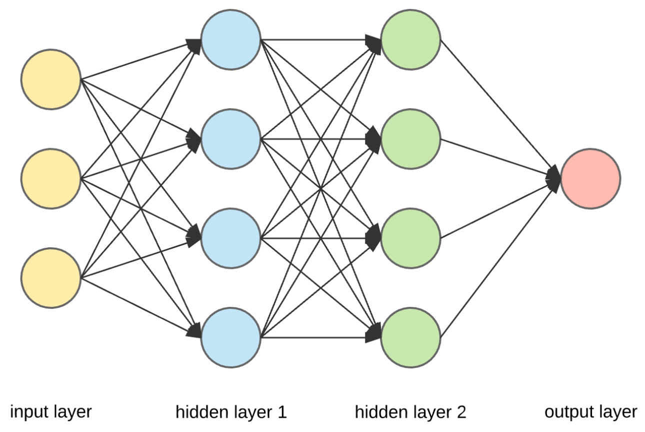


Fig. 5. Architecture of Fully Connected Neural Network

After the training process is done the encoder is saved, unlike the decode which is replaced by a fully connected neural network.

This fully connected neural network is separately trained (using back propagation) based on the output of the encoder and classification label in hand which will a supervised approach.

The data-sets used in this approaches are selected on

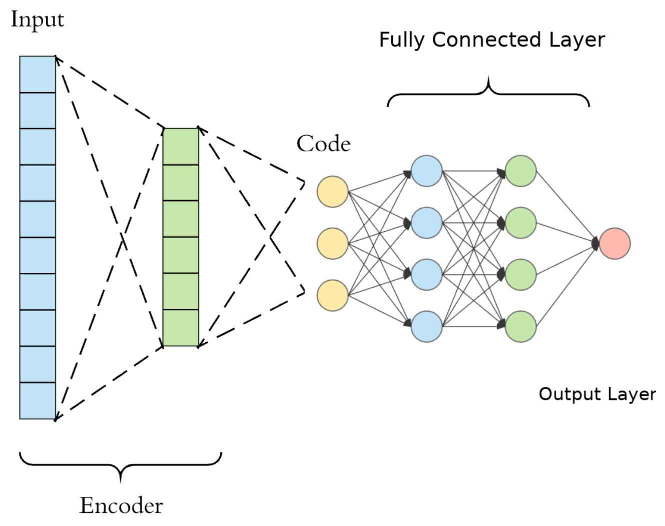


Fig. 6. Combined architecture used for auxiliary feature learning

the basis of how it represents the real world problems, such as limited instance of examples and moderate number of attributes to learn from. All of there are multivariable classification problems.

The data-set used to test this approach are as follows:

- Audit Data Data Set [Online] Available: http://archive.ics.uci.edu/ml/datasets/Audit+Data
- Wine Data Set [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Wine
- Breast Cancer Coimbra Data Set [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coim

Some explorations on the Breast Cancer Coimbra Data-set attributes are shown below:

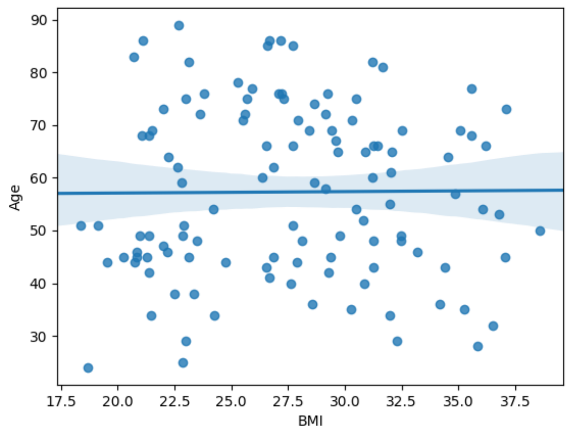


Fig. 7. Feature distribution between attribute Age and BMI

Scatter plot is used to compare the co-relation between two different features. Ideally features which are not co-related should be used for training. Checking whether there are any missing Values in the data-set is also important for

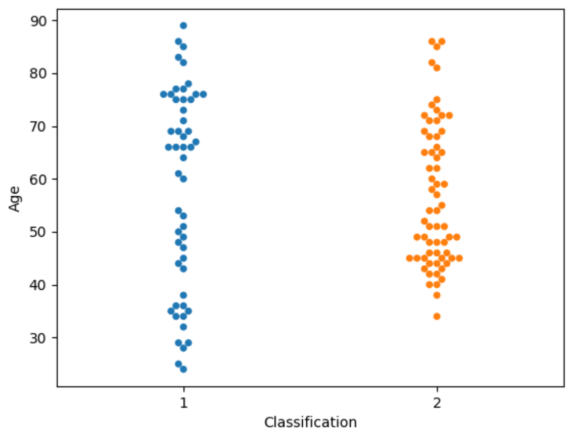


Fig. 8. Feature distribution for attribute Age

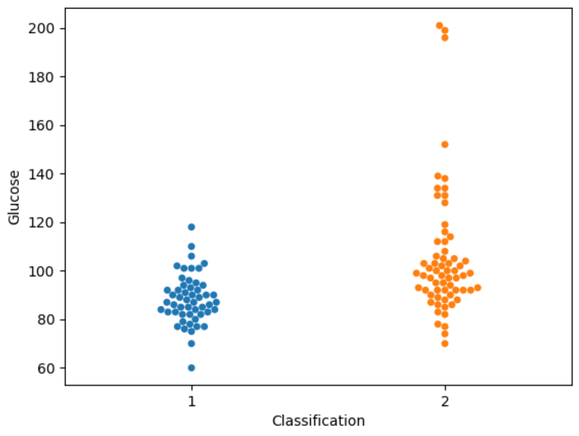


Fig. 9. Feature distribution for attribute Glucose (with outliers)

prepossessing step

Individual feature distribution are analyzed based on the class to check for outliers. Fig 9 shows signs of possible outliers in class 2, which can use remove during training to see if it helps in improving the performance of the system.

Finally, bar chart can be used to see the data-sets distribution based on the classes and check if it needs to be balance.

IV. EXPERIMENTS

As mentioned earlier, a small sample (X) from the entire dataset is used to train the Autoencoder, from which its output is sent to the fully connected layers for prediction. This arbitrary size (D) is increased in a iterative fashion (1)

$$D = X + n(C) \tag{1}$$

X - Initial sample size,
n - Number of training runs,
C - Constant increase in sample size

This is to test how the sample size used to train the Auto-encoder affects the overall performance of the model in generalization.

What is expected is an initial low performance of the model when n = 0, and upon increasing "n" there will be an increased performance and slowly diminishing in the gain for higher values of "n".

There is a number of hyper parameters that can be tweaked to find the optimal performance of the Auto-encoder.

Firstly, the number of stacked hidden layer used will play a significant role is the performance of the model.

Secondly, the number of neurons in each layer, more importantly on the end of the encoder which acts as a bottleneck for the information to pass through.

Thirdly, the learning rate and weight decay also plays an important role, which can lead to better results is tuned properly

The above-mentioned parameters also affect the fully connected neural network. which is responsible for the actual pattern recognition. Unlike the auto-encoders, the entire data-set can be used for training the fully connected neural network.

The number of epoch for all the models will be set to content so that there is a fair comparison between the different data-set sizes. An early stopping criteria is used to save the best models and stop if there is no significant improvement between x number of epochs.

V. DISCUSSION

The performance of the experiments can be evaluated by different metrics; Accuracy is one of the popular ones. Which might not necessarily give us the entire picture. For instance, having an unbalanced data-set where one class is predominately dominated say 95 percentage of the entire data-set, then a model that always predict the dominant class will have an accuracy of 95 percentage. Then accuracy is not a good evaluation metrics. In those cases, It is important to have more one metric to evaluate the performance of the model[40].

Precision and Recall helps in this case. Precision will indicate out of all the prediction the model made, how much times was it right. Recall on the other hand will indicate out of all the true label how many of them were retrieved. Depending on the application either Recall or Precision will be given more priority. Having one these matrices higher than the other doesn't mean they are better [18]. The challenge is to improve these metrics together. F1 is a good indication of these two metrics.

Formally F1 is indicated by:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Similarly, in Fig 10 Area under the ROC Curve (AUC) is another metric that help in case of binary classifier where the true positive rate (Sensitivity / Recall) vs false positive rate (Specificity) are plotted at a various threshold.

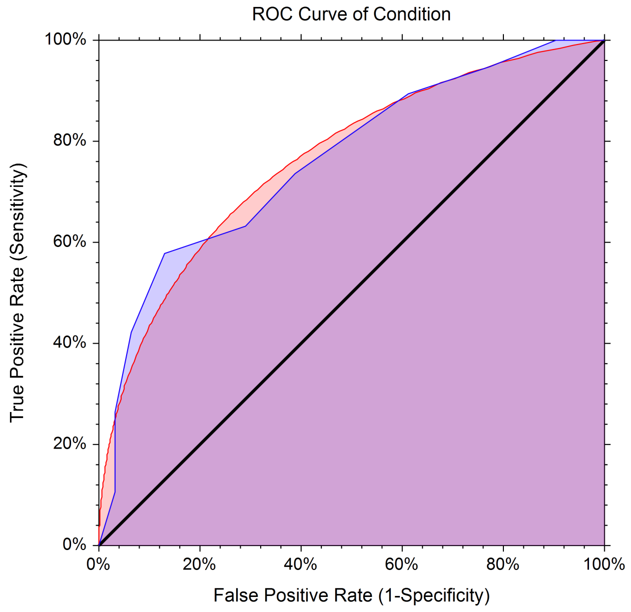
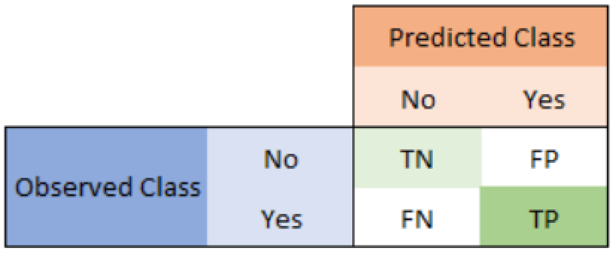


Fig. 10. Possible optimal ROC

Given the experiment is done on the basis of classification; Confusion matrix in Fig 11 is a good way to visually evaluate which of the classes are predicted collectively and which they are not. Ideal confusion matrix should be 0's except for the True Negative and True Positive.



		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

Fig. 11. Confusion Matrix

TN - True Negative
TP - True Positive
FN - False Negative
FP - False Positive

Accuracy = (TN + TP) / (TN + FP + FN + TP)
Precision = TP / (FP + TP)
Sensitivity = TP / (TP + FN)
Specificity = TN / (TN + FP)

If False Positive and False Negative are higher than acceptable range, further action should be taken to improve the model performance on those week classes.

Comparison between different tests mentioned in the above section will be on the ROC metrics. since it provides a more robust evaluation of the models [10].

VI. CONCLUSIONS

The approach presented here for helping the deep learning architecture learn auxiliary features has proven to have statistically significant improvements[1].

The cost of learning from the small data-set outweighs the cost of learning from huge data-sets. which is quite expensive to acquire and label them depending on the application.

REFERENCES

[1] Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization.

[2] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization.NIPS, 2000.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-samplingtechnique.Journal of artificial intelligence research, 16:321357, 2002.

[4] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmarkfor measuring progress in statistical language modeling.arXiv, 2013.

[5] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness toadversarial examples.ICML, 2017.

[6] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks.NIPS, 2017.

[7] T.DeVries and G. W. Taylor. Dataset augmentation in feature space.ICLR Workshops, 2017.

[8] H.Drucker and Y. Le Cun. Improving generalization performance using double backpropagation.IEEETransactions on Neural Networks, 3(6):991997, 1992.

[9] I.Goodfellow. Tutorial: Generative adversarial networks.NIPS, 2016.

[10] I.Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y.Bengio.Generative adversarial nets.NIPS, 2014.

[11] Liebel, Lukas, and Marco Krner. Auxiliary Tasks in Multi-task Learning.

[12] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet:A deep convolutional encoder-decoder architecture forimage segmentation.Transactions on Pattern Analysisand Machine Intelligence, 39(12):24812495, 2017.

[13] R. Caruana. Multitask learning: A knowledge-basedsource of inductive bias. InInternational Conferenceon Machine Learning, pages 4148, 1993.

[14] R. Caruana.Learning to learn, chapter Multitask Learning,pages 95133. 1998.

[15] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam.Rethinking atrous convolution for semantic image seg-mentation.arXiv:1706.05587, 2017.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, andA. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fullyconnected CRFs.Transactions on Pattern Analysis andMachine Intelligence, 40(4):834848, 2018.

[17] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, andH. Adam. Encoder-decoder with atrous separable convolu-tion for semantic image segmentation.arXiv:1802.02611,2018.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. En-zweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele.The Cityscapes dataset for semantic urban scene under-standing. InConference on Computer Vision and PatternRecognition, pages 32133223, 2016.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, andV. Koltun. CARLA: An open urban driving simulator.InAnnual Conference on Robot Learning, pages 116,2017.

[20] D. Eigen and R. Fergus. Predicting depth, surface nor-mals and semantic labels with a common multi-scale con-volutional architecture. InInternational Conference onComputer Vision, pages 26502658, 2015.

[21] D. Eigen, C. Puhrsch, and R. Fergus. Depth map predic-tion from a single image using a multi-scale deep network.InInternational Conference on Neural Information Pro-cessing Systems, pages 23662374, 2014.

[22] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worldsas proxy for multi-object tracking analysis. InConferenceon Computer Vision and Pattern Recognition, pages 43404349, 2016.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Visionmeets robotics: The KITTI dataset.International Journalof Robotics Research, 32(11):12311237, 2013.

[24] D. P. Kingma and J. Ba. Adam: A method for stochasticoptimization. InInternational Conference on LearningRepresentations, pages 115, 2015.

[25] Jaderberg, Max, et al. Reinforcement learning with unsupervised auxiliary tasks.

[26] George Konidaris and Andre S Barreto. Skill discovery in continuous reinforcement learning do-mains using skill chaining. InAdvances in Neural Information Processing Systems, pp. 10151023, 2009.

[27] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learn-ing.CoRR, abs/1609.05521, 2016.

[28] Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrentreinforcement learning: A hybrid approach.arXiv preprint arXiv:1509.03044, 2015.

[29] Long-Ji Lin and Tom M Mitchell. Memory approaches to reinforcement learning in non-markoviandomains. Technical report, Carnegie Mellon University, School of Computer Science, 1992.

[30] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Andrea Banino, Hubert Soyer, Andy Ballard, MishaDenil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell.Learning to navigate in complex environments. 2016.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wier-stra, and Martin Riedmiller. Playing atari with deep reinforcement learning. InNIPS Deep Learn-ing Workshop. 2013.

[32] Choi, Keunwoo, et al. Transfer learning for music classification and regression tasks.

[33] Relja Arandjelovi c and Andrew Zisserman. Look, lis-ten and learn.arXiv preprint arXiv:1705.08168, 2017.

[34] Yusuf Aytar, Carl Vondrick, and Antonio Torralba.Soundnet: Learning sound representations from unla-beled video. InAdvances in Neural Information Pro-cessing Systems, pages 892900, 2016.

[35] Babu Kaji Baniya, Joonwhoan Lee, and Ze-Nian Li.Audio feature reduction and analysis for automatic mu-sic genre classification. InSystems, Man and Cyber-netics (SMC), 2014 IEEE International Conference on,pages 457462. IEEE, 2014.

[36] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whit-man, and Paul Lamere. The million song dataset. InProceedings of the 12th International Society for Mu-sic Information Retrieval Conference, October 24-28,2011, Miami, Florida, pages 591596. University ofMiami, 2011.

[37] Joan Bruna and St ephane Mallat. Invariant scatteringconvolution networks.IEEE transactions on patternanalysis and machine intelligence, 35(8):18721886,2013.

[38] Keunwoo Choi, Gy orgy Fazekas, Kyunghyun Cho, andMark Sandler. The effects of noisy labels on deepconvolutional neural networks for music classification.arXiv:1706.02361, 2017.

[39] Keunwoo Choi, Gy orgy Fazekas, and Mark Sandler.Automatic tagging using deep convolutional neuralnetworks. InThe 17th International Society of Mu-sic Information Retrieval Conference, New York, USA.International Society of Music Information Retrieval,2016.

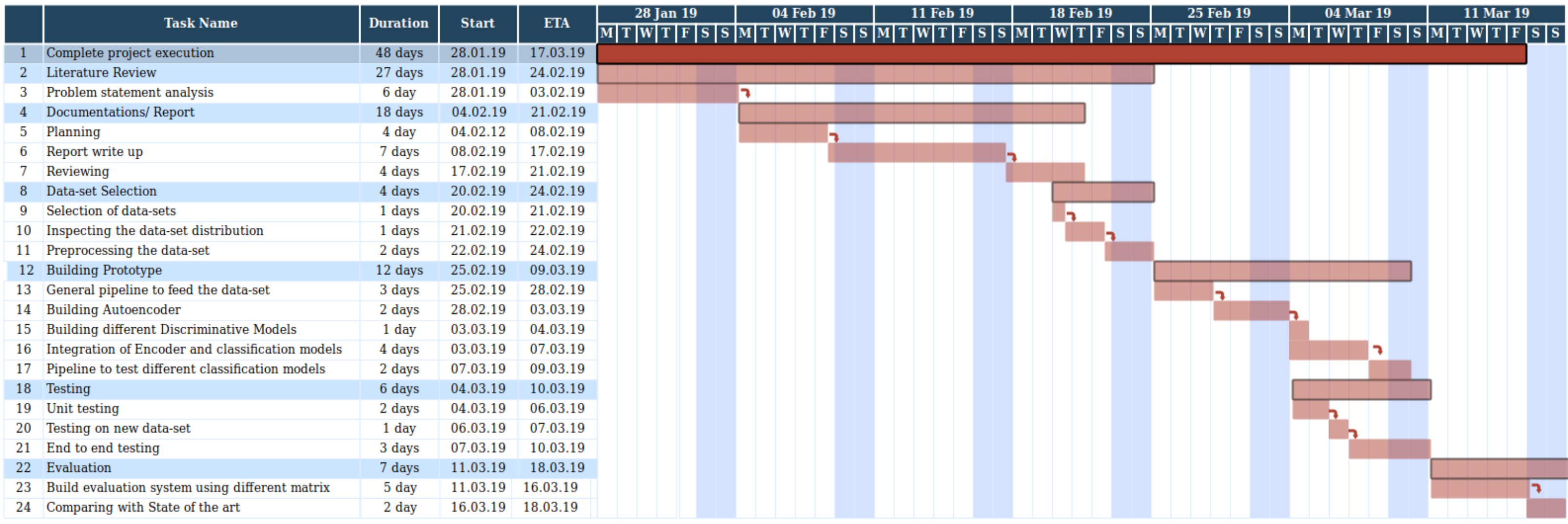
[40] Keunwoo Choi, Gy orgy Fazekas, and Mark Sandler.Explaining deep convolutional neural networks on mu-sic classification.arXiv preprint arXiv:1607.02444,2016.

[41] Keunwoo Choi, Gy orgy Fazekas, and Mark San-dler. Towards playlist generation algorithms using rnnstrained on within-track transitions. InWorkshop onSurprise, Opposition, and Obstruction in Adaptive andPersonalized Systems (SOAP), Halifax, Canada, 2016,2016.

VII. PLAN

- Initial literature survey was made to analysis the problem statement in the field of deep learning.
- Selection on data having chosen. few samples of data-sets to train the models on.
- We start off by building auto-encoder and different classifier models prototype.
- Training and Testing on new data-sets
- Testing the overall architecture end to end. and start with the experiment process
- Evaluating of result based on different metrics.
- Compare results with state of the art

Gan Chart for development cycle



	Task Name	Duration	Start	ETA
1	Complete project execution	48 days	28.01.19	17.03.19
2	Literature Review	27 days	28.01.19	24.02.19
3	Problem statement analysis	6 day	28.01.19	03.02.19
4	Documentations/ Report	18 days	04.02.19	21.02.19
5	Planning	4 day	04.02.12	08.02.19
6	Report write up	7 days	08.02.19	17.02.19
7	Reviewing	4 days	17.02.19	21.02.19
8	Data-set Selection	4 days	20.02.19	24.02.19
9	Selection of data-sets	1 days	20.02.19	21.02.19
10	Inspecting the data-set distribution	1 days	21.02.19	22.02.19
11	Preprocessing the data-set	2 days	22.02.19	24.02.19
12	Building Prototype	12 days	25.02.19	09.03.19
13	General pipeline to feed the data-set	3 days	25.02.19	28.02.19
14	Building Autoencoder	2 days	28.02.19	03.03.19
15	Building different Discriminative Models	1 day	03.03.19	04.03.19
16	Integration of Encoder and classification models	4 days	03.03.19	07.03.19
17	Pipeline to test different classification models	2 days	07.03.19	09.03.19
18	Testing	6 days	04.03.19	10.03.19
19	Unit testing	2 days	04.03.19	06.03.19
20	Testing on new data-set	1 day	06.03.19	07.03.19
21	End to end testing	3 days	07.03.19	10.03.19
22	Evaluation	7 days	11.03.19	18.03.19
23	Build evaluation system using different matrix	5 day	11.03.19	16.03.19
24	Comparing with State of the art	2 day	16.03.19	18.03.19