International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT21741

Deep Learning model to Automate the process of mapping Cancer Cells to Cell Lines & Cancer Types from Single Cell RNA-Seq Data

Vatsal Patel
BS Mathematics, St. Xavier's College, Mumbai, India

Article Info
Volume 7, Issue 4
Page Number: 17-26

Publication Issue :
July-August-2021

Article History
Accepted : 25 June 2021
Published : 02 July 2021

ABSTRACT

Single Cell RNA Sequencing has given us a broad domain to study heterogeneity & expression profiles of cells. Downstream analysis of such data has led us to important observation and classification of cell types. However, these approaches demand great exertion and effort added that it seems the only way to proceed ahead for the first time. Results of such verified analysis have led us to create labels from our dataset. We can use the same labeled data as an input to a neural network and this way we would be able to automate the tedious & time-consuming process of downstream analysis. In this paper, we have automated the process of mapping cancer cells to cancer cell lines & cancer types. For the same, we have used pan-cancer single cell sequencing data of 53513 cells from 198 cell lines reflecting 22 cancer types.

Keywords : Single Cell RNA-Seq, Deep Learning, Cancer Biology, Cancer Cell Lines, Automation, Neural Network, Gene Expression, Transfer Learning

I. INTRODUCTION

Transcriptional profiling of thousands of individual cells is possible with Single-Cell RNA-Seq. This degree of throughput analysis allows researchers to see what genes are expressed, in what quantities, and how they differ across thousands of cells in a heterogeneous sample at the single-cell level. Getting better insights into individual cell tissues has gotten a lot easier thanks to Single Cell RNA-Seq. As a result, more information and a better understanding of immunology and various diseases are available. In comparison to traditional procedures, this technology allows you to evaluate millions of single cells with high throughput in a cost-effective manner. Referring to our key paper "Pan-Cancer single cell RNA-seq identifies recurring programs of cellular heterogeneity[1]" has identified 12 expression programs that are recurrently heterogeneous within multiple cell lines. For the same, they had assigned profiled cells to 198 cancer cell lines reflecting 22 cancer types. They had assigned profiled cells to cell lines based on the consensus between two complementary approaches, which used genetic and expression profiles. In the first method, cells were clustered by their global expression profiles, and each cluster was mapped to the cell line with the most similar bulk RNA-seq profile[2]. In the Second method by detection of SNPs in the scRNA-seq reads, they assigned cells to the cell line with the highest similarity by SNP profiles derived from bulk RNA-seq[2-4]. These cell line assignments were consistent for 98% of the cells, they were used for the downstream analysis which led them to result in 12

expression programs as mentioned. As a result, they assigned cells to cell lines from single cell RNA-seq data, so now eventually we got a labeled data set where all the genes in the data are features and the respective cell line mapped is the label. As this cell line also reflects the cancer type we have another label as well. Such data can be an easy input to a neural network and can be trained to predict the labels with good accuracy. This way we would be automating the process of mapping cancer cells to cell lines. The Genes of each cell will be the features input to the model and the Cell Lines & Cancer types will be the labels i.e the outputs of the model. However, our model will be restricted to predict from those 198 cell lines reflecting 22 cancer types only but the layers for the same model can be used later on if one has to add additional cell lines by the method of transfer learning.

Traditional Approach Followed by Authors of our key paper to Classify Cells to cell lines & Cancer types.

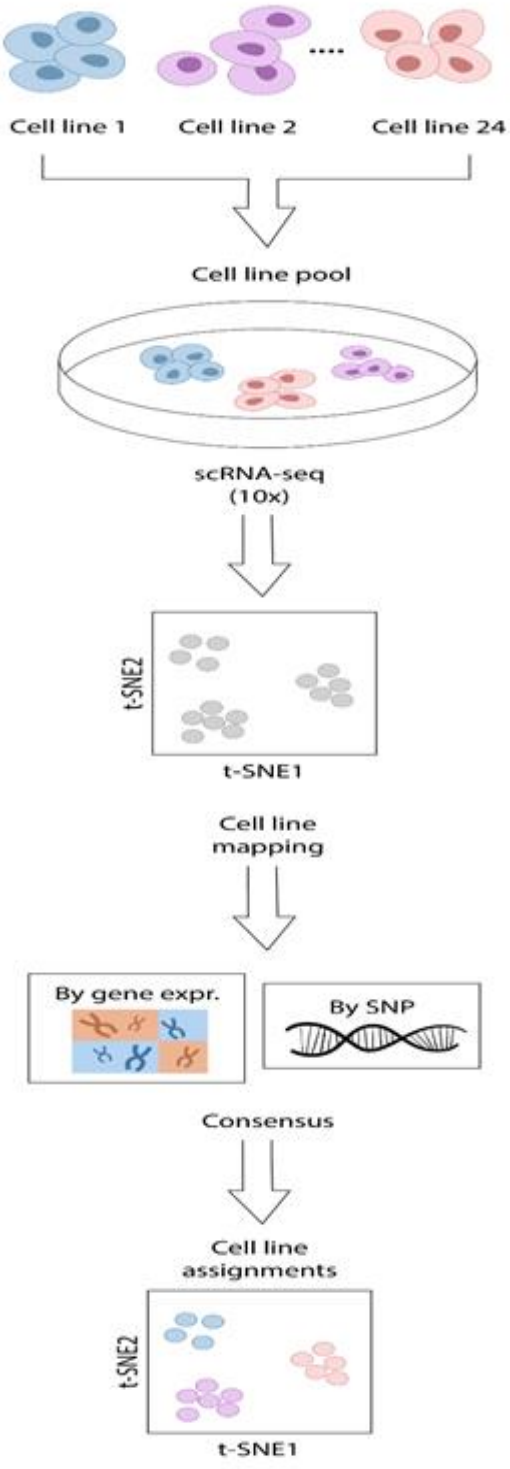


Fig-1

II. Method & Discussion

Single Cell RNA - seq data is of the form Genes X Cells, the authors of our key paper1 had already preprocessed the data & done the quality control

work. The resultant dataset obtained post cells mapped to cell lines was of the form Cell_Line X Genes X Cells. I used the same data as my raw data. There were in total 198 Cell Lines, 31015 Genes & 53513 Cells. These 198 Cell lines reflected 22 cancer types as mentioned above. Both Cell lines and Cancer type are labels for us.

2.1 Pre-Processing

Observing the structure of the data it is easy to notice that there is an unequal distribution of the number of cells with respect to each cell line. So taking a random split of 60-20-20 for training-validation-test data directly on the whole dataset might not be a feasible approach as we might end up having more cells from one cell line in the training set while fewer cells from other cell line and this way we might end up creating a model that might be better at predicting one cell line than other. So we need to split up the data such that there is an even distribution across the labels for training the model. At the same time, we would also face multiclass label loss distribution but it will be covered up ahead in the model by the softmax function of Keras.

To keep an intuition of the data structure imagine it as a three-dimensional matrix with each layer from top representing one cell line and this layer is of the form Genes X Cells. ASo we started selecting each layer then taking a transpose of the layer and added two new columns in the layer denoting as 'Cell_Line' and 'Cancer_Type' filling it with the string names of cell line and cancer type it reflected respectively for all the cells in that layer. Then we had a random split of 60-20-20 for the train-validation-test sets of each layer. Then clubbed all the train, validation, test sets for each layer denoted now as TRAIN, TEST, VALIDATE respectively, and shuffled the rows randomly. However we need to normalize this data, so for that we will also club TRAIN, TEST & VALIDATE which will be in the matrix of form Genes X Cells. Further on we took transpose of this clubbed data as it makes it easier for us to work with, so now we had it of the structure Cells X Genes denoted as DATA in our paper.

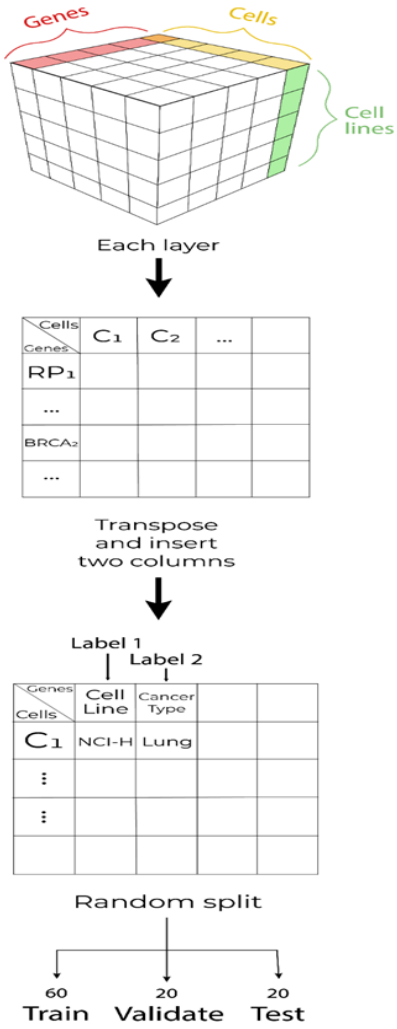
First, we separated the two label columns which are 'Cell_Line' & 'Cancer_Type' from TRAIN, TEST & VALIDATE. As these labels are in string format we need to convert them into label forms that can be accepted by the neural network. So for that we first created two python dictionaries and saved unique values from those two label columns as key and unique iterated integer as value. Label structures are represented below in Table 1. Now that we have integer-coded label values we replaced the values in the labels column with the relevant integer. Further on we one-hot encoded this labels column as it is much better for the loss function of the TensorFlow API to work with.

Then we used two different normalization techniques (1) Min-Max Scaling and (2) Z-Score normalization. For (1) Min-Max Scaling we calculated the max and min value with respect to each column of DATA and then applied the formula of Min-Max Scaling on individual data frames of train, validation, test set and to be denoted as train_s, validation_s, test_s respectively. For (2) Z-Score Normalization we calculated the mean and standard deviation with respect to each column of DATA and then applied the formula of Z-Score normalization on individual data frames of train, validation, test set and to be denoted as train_z, validation_z, test_z respectively. Now we are ready to input this into a neural network.

2.2 Selection of an Optimal Neural Network for Classification of Cancer Cells to Cancer Cell lines

We started by creating a basic neural network with 4 dense layers of neurons structured as 32,64,64,198 we denote it as model 1 using Keras API in TensorFlow. We also had to implement L2 kernel regularizers at each of these hidden layers. Where the 198 neurons

in the output layer resemble to classify for the cell lines labels, it has the activation function softmax of Keras API from TensorFlow. We instantiated two copies of model 1 and trained one with the Z-Score normalized data while the other with Min-Max normalized data. Taking a look at the training curve of accuracy and loss for training and validation, it becomes quite clear that Min-Max normalized data suits better for the task as the one with Z-Score normalized shows a lot of aberrations, check out Table 1 for the same. We can argue at this point that the Z-Score data has gene-expression values negative for certain genes and Fig-2 part 1



Follow Fig-2 part 2 from this step

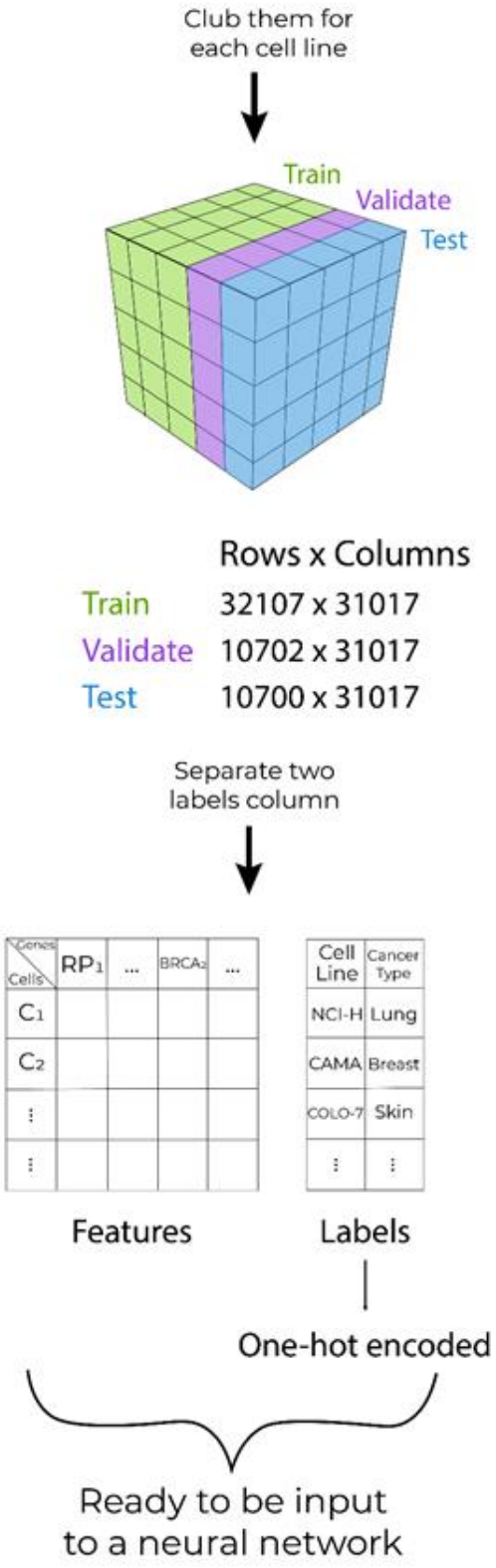


Fig-2 Part 2

Graphical Representation of Pre-Processing. Note down that the raw data we used was of the obtained

to us as a part of the process performed in the key paper.

cells, so it kind of makes no sense to how you would interpret a gene expression of a cell at that moment as negative instead of it being zero. Hence it is much better to use Min-Max normalized data.

Then we created 4 models with different specs represented in Table 2 and trained them with Min-Max normalized data for 30 epochs as well as plotted the accuracy & loss curves along with evaluating the test data to interpret which one suits better, check out Table 3 & 4 for the same. Our one goal in mind about selection is that the model should not have an unnecessary amount of parameters as it would just increase the computational load on the machine, so we would like to approach with a model that is easy to publish and can be loaded on a simple machine as well. After the interpretation, it was quite clear that Model 2 seems fine enough to work with our data added that we have a benefit of large amounts of data for training in the first place i.e around 32107 cells. Also the trade-off for selecting Model 2 then 1 is that Model 1 might be overfitting the data at the end. In practice, we had tried with these 4 models having a varied number of neurons per layer as well but these 4 models were the prime structures we tried out working with represented in Table 2.

Further to train our neural network better we approached by giving the model batched data as an input, for the same we tried with different sizes of the batch for a specific number of epochs and figure out which one suits them best, one can interpret the results from Table 5. It seems feasible to select the number of epochs as 20 and batch size as 128. That way we will also not be overfitting the model. Now we can implement checkpoints for our model & also save the weights of the model so we can use it later on for our use. While running through all of them we had always tested our model at the end of the test data and the results are also mentioned in Table 4.

2.3 Selection of Optimal Neural Network for Classification of Cancer Types

As we have already created a model to classify 198 cell lines, now we need to make one that can classify cancer cells to 22 cancer types reflected from these 198 cell lines. So for that we approached with the idea of transfer learning. As we have already created a model that can classify 198 cell lines we can use the same for 22 cancer types. We loaded the trained model to classify 198 cell lines along with the learned weights and then added a layer of 22 neurons with activation function softmax at the end of the previous model. Then trained this model with as low as 6 epochs and gained an accuracy of 0.97 & a loss of 0.45 on the test set. This should be fine enough for us as it won't be overfitting the data if we seek out more accuracy. Saved this model with weights so one can use the same to do this process.

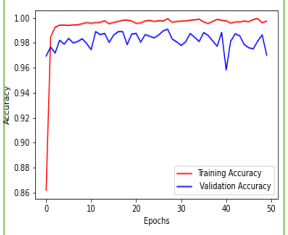
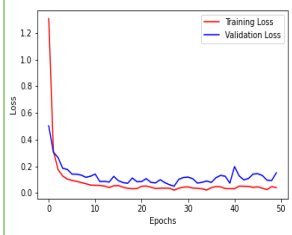
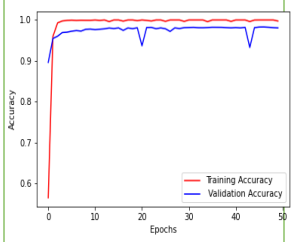
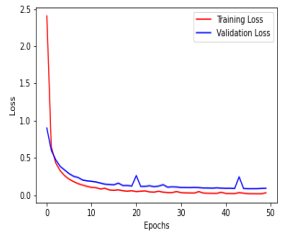
Check out the supplementary content files to see which 198 cell lines and 22 cancer types the two models can predict & one should make sure that while using the model the input vector to the model must have the same order of features i.e the genes.

III. Results & Observations

Here we presented all the tables related to our method from section 2, and it's quite clear to make a catch why we selected certain specific models.

Table 1

	Accuracy	Loss

Z-Score Normlaized		
Min-Max Normalized		

Plotting the training curves for Model 1 with Z-Score normalized & Min-Max Normalized data. The Red line represents training data and the blue one for validation data. It can be noticed that while training the model with Z-Score normalized data it has a lot of aberrations which is not a good sign.

Table 2 – Different Model Structures

```
Model: "Model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
d1_32 (Dense)                (None, 32)                992512
_________________________________________________________________
d2_64 (Dense)                (None, 64)                2112
_________________________________________________________________
d3_64 (Dense)                (None, 64)                4160
_________________________________________________________________
output (Dense)               (None, 198)               12870
=================================================================
Total params: 1,011,654
Trainable params: 1,011,654
Non-trainable params: 0
```

```
Model: "Model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
d1_32 (Dense)                (None, 32)                992512
_________________________________________________________________
d2_64 (Dense)                (None, 64)                2112
_________________________________________________________________
Dropout1_0.2 (Dropout)       (None, 64)                0
_________________________________________________________________
d3_64 (Dense)                (None, 64)                4160
_________________________________________________________________
Dropout2_0.2 (Dropout)       (None, 64)                0
_________________________________________________________________
output (Dense)               (None, 198)               12870
=================================================================
Total params: 1,011,654
Trainable params: 1,011,654
Non-trainable params: 0
```

```
Model: "Model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
d1_32 (Dense)                (None, 32)                992512
_________________________________________________________________
d2_64 (Dense)                (None, 64)                2112
_________________________________________________________________
Dropout1_0.2 (Dropout)       (None, 64)                0
_________________________________________________________________
d3_64 (Dense)                (None, 128)               8320
_________________________________________________________________
Dropout2_0.2 (Dropout)       (None, 128)               0
_________________________________________________________________
d4_64 (Dense)                (None, 64)                8256
_________________________________________________________________
Dropout3_0.2 (Dropout)       (None, 64)                0
_________________________________________________________________
output (Dense)               (None, 198)               12870
=================================================================
Total params: 1,024,070
Trainable params: 1,024,070
Non-trainable params: 0
```

```
Model: "Model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
d1_32 (Dense)                (None, 32)                992512
_________________________________________________________________
d2_64 (Dense)                (None, 64)                2112
_________________________________________________________________
d3_64 (Dense)                (None, 128)               8320
_________________________________________________________________
d4_64 (Dense)                (None, 64)                8256
_________________________________________________________________
output (Dense)               (None, 198)               12870
=================================================================
Total params: 1,024,070
Trainable params: 1,024,070
Non-trainable params: 0
```

The structure of the neural network of the 4 models we tried out added that we had tried with a varied number of neurons in the hidden layers initially but in practice, we chose this once. It was build using Keras API in TensorFlow. Also to note that we have used L2 kernel regularizers at each of those hidden layers.

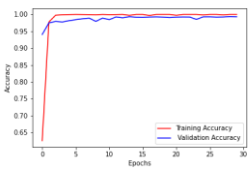
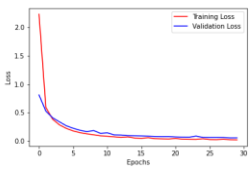
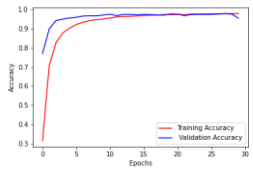
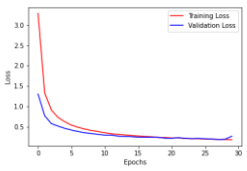
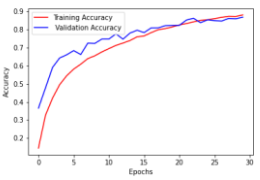
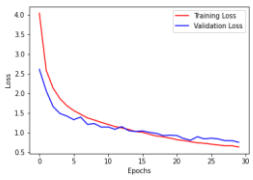
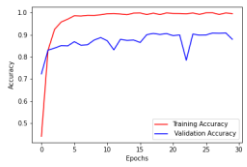
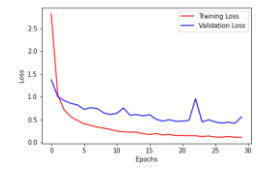
Table 3

Results of 4 different models on the test set. We chose Model_2 as it would be the best candidate

which will not overfit as well as it has an acceptable accuracy

Model Names	Epochs	Test Loss	Test Accuracy
Model_1	30	0.0572	0.9927
Model_2	30	0.2520	0.9520
Model_3	30	0.7626	0.8639
Model_4	30	0.5470	0.8796

Table 4 - Different Model Training Curves

	Accuracy	Loss
Model_1		
Model_2		
Model_3		
Model_4		

Training curves of all 4 Models. We chose to work with model 2 as it doesn't show any aberrations in the training curves for 30 epochs. Further, we trained model 2 with batches which would even optimize training more.

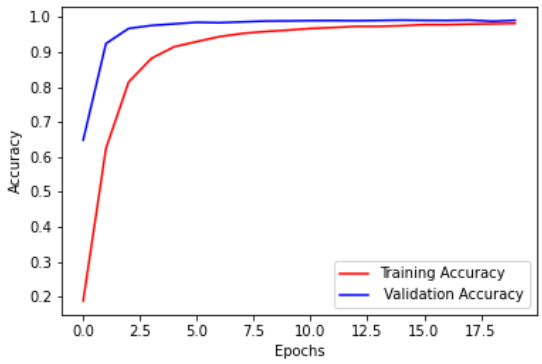
Table 5

Model 2 Types	Epochs	Batch Size	Test Loss	Test Accuracy
K1	10	16	0.4807	0.9202
K2	10	32	0.4634	0.9328
K3	10	256	0.4560	0.9872
K4	10	512	0.5817	0.9796
K5	20	32	0.4800	0.9136
K6	20	128	0.2729	0.9836
K7	20	512	0.3872	0.9889
K8	25	256	0.2391	0.9936

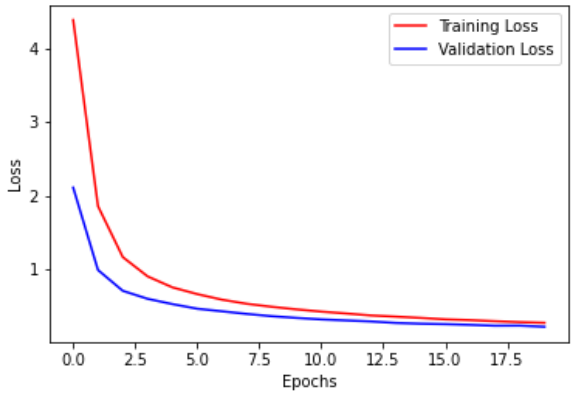
Table 5 gives a review on how Model 2 works with different

Table 6- Final Model for prediction of Cancer Cell Line

Accuracy



Loss



Training Curves for Model to predict Cancer Cell Line. It has Test loss: 0.2124 & Test accuracy: 0.9932.

We call this model CancerCellLinePredictor. As discussed above we added a new layer of 22 neurons at the end for classifying cancer types on the same model and we denote the model as CancerTypePredictor. It has a Test loss: 0.4190 & Test accuracy: 0.9915.

Training Curves for Model to predict Cancer Cell Line. It has Test loss: 0.2124 & Test accuracy: 0.9932. We call this model CancerCellLinePredictor. As discussed above we added a new layer of 22 neurons at the end for classifying cancer types on the same model and we denote the model as CancerTypePredictor. It has a Test loss: 0.4190 & Test accuracy: 0.9915.

One should keep a note that when using the model to predict they should give the input feature vector having the same gene names in the order and it is Min-Max normalized data.

IV. Conclusion

We can conclude that it is possible to automate traditional & tedious approaches very easily with the help of machine learning. One can now use this model if they have an interest to map cancer cells to cell lines or cancer types in particular of related our set. We have saved the model with weights and saved it as CancerCellLinePredictor & CancerTypePredictor in the code files. Also this model can also be used to classify a cell line or cancer type which is not present in our data by using the approach of transfer learning. We have had an advantage of training the model easily because of a large amount of data added that is Pan-Cancer data which help also us to conclude that it is a generalized model.

V. Limitations & Future Scope

Currently, the model is limited to 198 cell lines and 22 cancer type but as mentioned above by the method of transfer learning one can keep adding additional neurons in the last layer and train the model a few epochs on the saved weights of the current model & would again gain the required results.

VI. Data Availability

Raw and processed scRNA-seq data are available through the Broad Institute's single-cell portal (SCP542) and at the Gene Expression Omnibus (GEO) (accession number GSE157220). Publicly available databases used in our analysis included the DepMap portal (18q3 data release; https://depmap.org/), the CCLE portal (https://portals.broadinstitute.org/ccle),

the CTD2 portal (https://ocg.cancer.gov/programs/), GTRD database version 20.06 (http://gtrd.biouml.org) and MSigDB version 7.0 (https://www.gsea-msigdb.org/gsea/msigdb/index.jsp).

We Used the Data from the results of the experiment performed in our key paper located in CCLE_heterogeneity_Rfiles as CCLE_scRNAseq_CPM.RDS obtained after extraction of the zip file CCLE_scRNAseq_github.zip obtained from Broad Institute's single-cell portal.

VII. Code availability

Code for all steps from downloading the data to pre-processing to training the deep learning model is available at https://github.com/Science1804/Deep-Learning-model-to-Automate-the-process-of-mapping-Cancer-Cells-to-Cell-Lines-Cancer-Types.
The code file to run the saved model to predict Cancer Cells and Cancer Type is available here as "Using the model to Predict Cancer Cell lines and Cancer Types.ipynb".

VIII. Acknowledgements

I would like to thank my family and friends for the love & support they have given me. I would also like to thank Prof. Nidhan H Biswas from NIBMG, West Bengal, India for forwarding me the key paper which led me to the idea of automating the process of mapping cells to cell lines using Deep Learning.
I would also like to thank my colleague Pearl D'Souza for creating the graphical abstract of Fig-1,2.

IX. Supplementary Content

All the Tables related to the paper & an additional Training curves table for choosing model 2 with different epochs and batch size are attached at https://github.com/Science1804/Deep-Learning-model-to-Automate-the-process-of-mapping-Cancer-Cells-to-Cell-Lines-Cancer-Types.

X. REFERENCES

[1]. Kinker, G.S., Greenwald, A.C., Tal, R. et al. Pan-cancer single-cell RNA-seq identifies recurring programs of cellular heterogeneity. Nat Genet 52, 1208–1218(2020).https://doi.org/10.1038/s41588-020-00726-6

[2]. Ghandi, M., Huang, F.W., Jané-Valbuena, J. et al. Next-generation characterization of the Cancer Cell Line Encyclopedia. Nature 569, 503–508 (2019). https://doi.org/10.1038/s41586-019-1186-315

[3]. Yu, C., Mannan, A., Yvone, G. et al. High-throughput identification of genotype-specific cancer vulnerabilities in mixtures of barcoded tumor cell lines. NatBiotechnol 34, 419–423 (2016) https://doi.org/10.1038/nbt.3460

[4]. Kang, H., Subramaniam, M., Targ, S. et al. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. Nat Biotechnol 36, 89–94 (2018). https://doi.org/10.1038/nbt.4042

[5]. Ainscough, B.J., Barnell, E.K., Ronning, P. et al. A deep learning approach to automate refinement of somatic variant calling from cancer sequencing data. Nat Genet 50, 1735–1743 (2018). https://doi.org/10.1038/s41588-018-0257-y

[6]. Van Valen DA, Kudo T, Lane KM, Macklin DN, Quach NT, DeFelice MM, et al. (2016) Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments. PLoS Comput Biol 12(11): e1005177. https://doi.org/10.1371/journal.pcbi.1005177

[7]. Mets DG, Brainard MS (2018) An automated approach to the quantitation of vocalizations and vocal learning in the songbird. PLoS Comput Biol 14(8):

e1006437.https://doi.org/10.1371/journal.pcbi.1006437

[8]. Frasier KE, Roch MA, Soldevilla MS, Wiggins SM, Garrison LP, Hildebrand JA (2017) Automated classification of dolphin echolocation click types from the Gulf of Mexico. PLoS Comput Biol 13(12): e1005823.https://doi.org/10.1371/journal.pcbi.1005823

[9]. Roman T, Xie L, Schwartz R (2017) Automated deconvolution of structured mixtures from heterogeneous tumor genomic data. PLoS Comput Biol 13(10): e1005815. https://doi.org/10.1371/journal.pcbi.1005815

[10]. Sekar JAP, Tapia J-J, Faeder JR (2017) Automated visualization of rule-based models. PLoS Comput Biol 13(11): e1005857. https://doi.org/10.1371/journal.pcbi.1005857

[11]. C. Petschnigg, S. Bartscher and J. Pilz, "Point Based Deep Learning to Automate Automotive Assembly Simulation Model Generation with Respect to the Digital Factory," 2020 9th International Conference on Industrial Technology and Management (ICITM), 2020, pp. 96-101, https://doi.org/10.1109/ICITM48982.2020.9080347

[12]. Hammad, Issam, R. Simpson, H. D. Tsague and Sarah Hall. “Using Deep Learning to Automate the Detection of Flaws in Nuclear Fuel Channel UT Scans.” ArXiv abs/2102.13635 (2021): n. Pag.

[13]. Zampieri G, Vijayakumar S, Yaneske E, Angione C (2019) Machine and deep learning meet genome-scale metabolic modeling. PLoS Comput Biol 15(7): e1007084.https://doi.org/10.1371/journal.pcbi.1007084

[14]. Tyson AL, Rousseau CV, Niedworok CJ, Keshavarzi S, Tsitoura C, Cossell L, et al. (2021) A deep learning algorithm for 3D cell detection in whole mouse brain image datasets. PLoS Comput Biol 17(5): e1009074.https://doi.org/10.1371/journal.pcbi.1009074

[15]. Amarasinghe Kaushalya C., Lopes Jamie, Beraldo Julian, Kiss Nicole, et al. (2021) A Deep Learning Model to Automate Skeletal Muscle Area Measurement on Computed Tomography Images. Frontiers in Oncology: https://doi.org/10.3389/fonc.2021.580806

[16]. Lugagne J-B, Lin H, Dunlop MJ (2020) DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. PLoS Comput Biol 16(4): e1007673. https://doi.org/10.1371/journal.pcbi.1007673

[17]. Ted Spaide,Yue Wu,Ryan T. Yanagihara, et al. (2020) Using Deep Learning to Automate Goldmann Applanation Tonometry Readings : American Academy of ophthalmology DOI:https://doi.org/10.1016/j.ophtha.2020.04.033

[18]. Evan J. Zucker,Zachary A. Barnes,Matthew P. Lungren et al. (2019) Deep learning to automate Brasfield chest radiographic scoring for cystic fibrosis : American Academy of ophthalmology DOI:https://doi.org/10.1016/j.jcf.2019.04.016

Cite this article as :

Vatsal Patel, "Deep Learning model to Automate the process of mapping Cancer Cells to Cell Lines & Cancer Types from Single Cell RNA-Seq Data", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 4, pp. 17-26, July-August 2021. Available at doi : https://doi.org/10.32628/CSEIT21741
Journal URL : https://ijsrcseit.com/CSEIT21741