# Stacked Autoencoder and Mutual Information Criterion Affect on MLP and CNN Classification

Rafe Kruse
*Herbert Wertheim College of Engineering*
*University of Florida*
Gainesville, USA
rafekruse@ufl.edu

Bo Chen
*Herbert Wertheim College of Engineering*
*University of Florida*
Gainesville, USA
bo.chen@ufl.edu

Contributions
Rafe Kruse: Part I, and corresponding report content
Bo Chen: Part II, and corresponding report content
Collaborative: MAP, Abstract, introduction, Discussion, Conclusion

***Abstract*——*In this project report, we present a comprehensive comparison study amongst four popular machine learning algorithms. First, we examine convolutional neural networks (CNN) for original data as a baseline classifier. Then, we compare the performance of the multilayer perceptron (MLP) and CNN for reduced dimensional data. Besides, MLP with quadratic mutual information (QMI) criterion is included as well. In order to quantify the performance of this classifier, maximum a posterior (MAP) is used to project the output of MLP to labels. As a result, the CNN classifier feeding in original data points can achieve test accuracy of around 91.8%. On the other hand, the MLP model using reduced-dimension data demonstrated about 87.63% on the testing set. Besides, the CNN with reduced-dimension data achieves 88.4% accuracy. In the end, the MLP with QMI criterion has 72.06% accuracy on the testing set.***

## I. INTRODUCTION

A number of studies have applied machine learning models, especially Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN), in a considerate number of tasks. These include but not limited to image classification[1][2], natural language processing[3], speech recognition[4][5], and text classification[6]. A MLP is a perceptron that teams up with additional perceptrons, stacked in several layers, to solve complex problems. CNN is a natural extension to MLP with few modifications. Mainly, the MLP algebraic dot product as a similarity function was replaced with two-dimension convolution; in addition to a pooling layer which reduces parameter dimensions making the model equi-variant to translations, distortions, and transformations. The sparse connectivity nature of CNN is also a variation to the MLP[7]. Before feeding data into a network, a dimension-reduced technique could be used for effective training. This is because for data with high dimensionality, only a few dimensions is useful for discrimination. One of the well-known techniques is stacked autoencoder (SAE).

However, in most of the state-of-art literatures, the number of units in the output layer has to be in accordance with the number of classes. Instead of using regular loss function like mean square error (MSE) or cross entropy criterion, quadratic mutual information (QMI) not only do not require same dimensions with labels, but also use the information of entire density instead of second order moment.

In this project report, we show a comparison study amongst CNN over original data, MLP for reduced dimensional data as well as MLP with quadratic mutual information criterion over original data. The objective of this project is to find which classifier gives the best performance. The rest of this report is organized as follows. Section 2 describes the approaches we use to construct each classifier and how we pick the set of hyperparameters with best performance. Section 3 presents the testing result of each classifier. The performance discussion and concluding remarks are included in section 4 and 5 respectively.

## II. METHODS

We explore the impact of utilizing an autoencoder network to reduce feature dimensionality for classification problems. Our stacked autoencoder was used to transform the 60000 training samples and 10000 test samples from the Fashion-MNIST[9] into a smaller 64 feature space for classification by our two MLP , one CNN, and one MAP classifiers.

### *2.1 Stacked Auto Encoder*

Stacked autoencoders (SAE) are extensions of the traditional sparse autoencoder which takes an input vector and attempts to reduce the dimensionality of the data to a more compressed representation(encoding) of the data. Autoencoders consist of two parts, encoder and decoder, while the encoder learns to reduce the feature space of the input data the decoder creates attempts to recreate the original input using the encoder's compressed representation of the data. Ideally this process will remove signal “noise” while also reducing the complexity of the learning problem.

Stacked autoencoders consist of layers of sparse autoencoders that feed the reduced representations from one hidden layer to next. These models are trained in a greedy layer-wise manner[8]. This means each layer of the stacked autoencoder is trained individually using the previous autoencoder’s embedded data as the new input and decoder output objective. Each layer is trained sequentially until getting to the bottleneck layer, the final layer before the decoder. After training the decoder portion of the model can be removed and the bottleneck layer of the encoder is used as input for classification models.

For our autoencoder we used the architecture shown in figure 1 with 500-200-64-200-500 hidden units. For selecting the number of units in the bottleneck layer we compared the reconstruction error as well as the relative

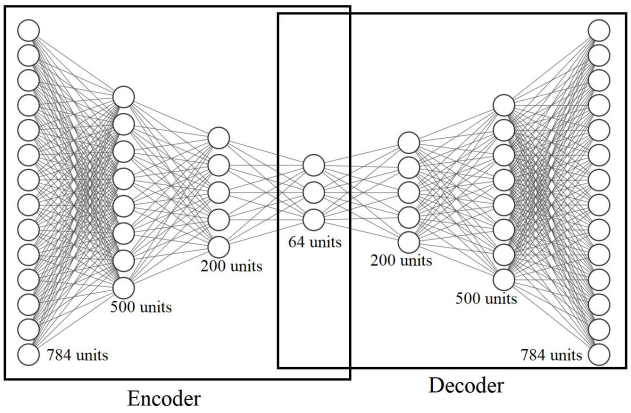


Figure 1. Stacked Autoencoder Architecture, the central 64 unit hidden layer is the overlap where the encoder outputs.

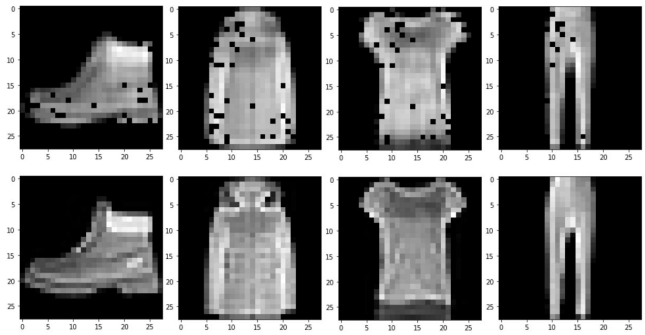

Figure 2. These are the reconstructed images output from the fully connected SAE. The upper images show the output of the decoder in image format and the corresponding ground truth image below.

performance on the MLP and CNN classifier. We found that while reconstruction error gave a good indication of how well the autoencoder was able to approximate the input image, some of the random initializations resulted in compressed features that underperformed when fed into the classifiers. This is likely due to the autoencoder learning to recreate the image well but not capturing feature information more relevant to classification. Because of this we chose to use the accuracy on the two models as our metric for selecting the ideal bottleneck size.

Our autoencoder uses fully connected linear layers with the rectified linear unit (ReLU) activation function. Our autoencoder was trained using a mean square error criterion, the ADAM[10] optimizer. We made use of mini batches of 200 samples that were chosen stochastically as they were ideal for approximating online learning while keeping a far shorter training time. This model was trained for 200 epochs with a learning rate of 0.0002 as any further epochs showed no improvement in recreation accuracy. The training was carried out using Pytorch's CUDA implementation to utilize a single GTX 1080 Ti resulting in a training time of 4 minutes and 37 seconds.

*2.2 Cross-Entropy Loss MLP Classifier*

Our first multilayer perceptron (MLP) model was constructed to make use of the 64 unit bottleneck layer of the SAE. This model was made up of 3 hidden layers with 128, 256, and 64 units respectively. The final output layer is 10 units corresponding to the 10 classes within the dataset we used. Each hidden layer linear unit used a ReLU activation function and the final output was put through a log softmax. This architecture was chosen using a grid search method and performed best with large networks showing no reduction in error rate. The ADAM optimizer was chosen because of it's proven success when training MLPs and the learning rate was set to 0.001. Similar to the architecture these hyper parameters were selected from a grid search of different models.

The MLP was trained similarly to non-encoded input MLPs using mini batches of 200 samples and fixed cross validation of the model after each epoch. Once the loss on the validation set degrades five times, we assume that the model starts memorizing the training data and overfitting has begun to occur. At this point is when we cease training of the model.

*2.3 Convolution Neural Network Classifier*

Within paper we have two relevant CNNs that are used for comparison. The first being the CNN trained for and described in [11] as well as a CNN trained using the output from our previously described SAE. It is the specifics of this model that we will describe further.

While the hyperparameter values were maximized through grid search similarly to the MLP, the architecture of a CNN can be far more varied. We opted to reference popular practice for the architecture of the CNN[12]. The network starts with two 1d convolution layers with a kernel size of 3 and 20 and 40 output channels respectively. In CNNs often there is a trade off of more convolutional layers being able to better generalize at the cost of computation time. We chose two layers because further layers show greatly diminishing returns and two allows for testing given our available hardware. The convolutions are followed by a single dropout layer and finally three fully connected linear units with 128, 256, and 64 units. The ReLU activation function was used at all units within the network with the final output being passed through the log softmax function.

The ideal result utilized the ADAM optimizer and was trained using mini-batches of 200 samples across 200 epochs with a learning rate of 0.002.

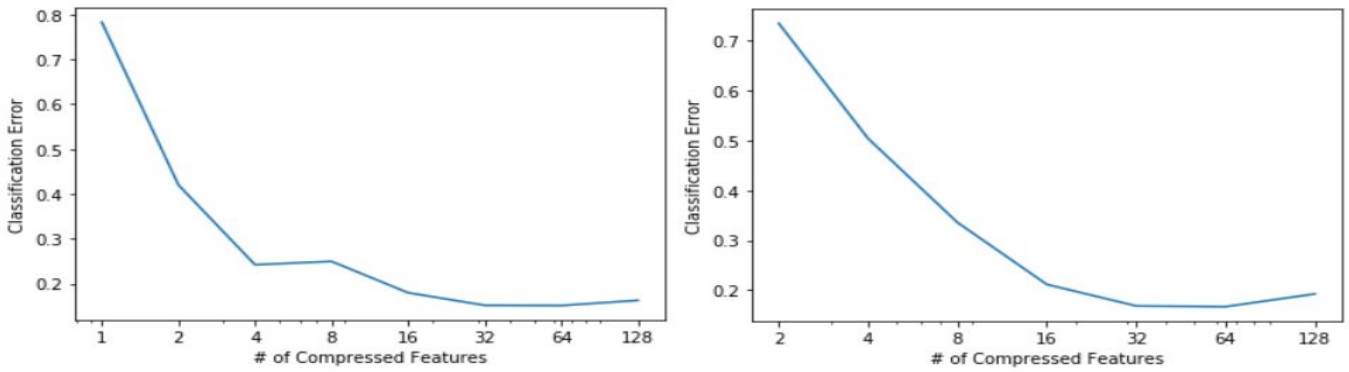


Figure 3: Comparison of MLP(left) and CNN(right) error on test data compressed by the above SAE with a bottleneck of between 2 and 128 features

*2.4 QMI MLP Classifier*

In this project, we also examine the performance of a MLP with quadratic mutual information (QMI) criterion. Compared to other state-of-art criterion used in deep/machine learning, QMI doesn't form the error signal which would give more flexibility to the designers. In other words, the outputs of the networks are not necessary to have the same dimensions with labels. Moreover, the outputs from different classes are not only uncorrelated, but independent. As a result, the QMI criterion can be used for many different applications. Particularly, we examine its performance in feature extraction in this report.

From the section above, we know that 64 units in the bottleneck layer would give best performance for feature extraction. Therefore here we set 64 units for the output layer. The structure is exactly the same with the encoder part of the autoencoder described above.

For faster convergence purposes, the optimizer of MLP is set to ADAM whose step size is adaptive and is bigger in the few epochs and keeps decreasing when epoch getting larger. The activation functions of the hidden layer

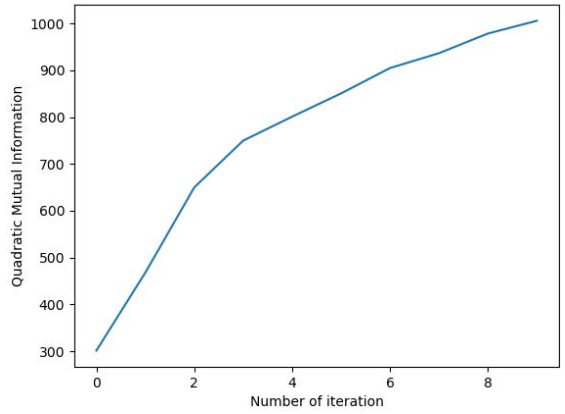


Figure 4: Estimated quadratic mutual information with respect to training epoch

are set to rectified linear activation functions (ReLU). Sigmoid function is used for units of output layer.

We use Gaussian kernel for density estimation with kernel size selected to 0.0005 based on Silverman's rule. The mini-batch size is set to 200 for best performance. The mutual information plot with respect to epoch can be seen in figure 4.

The training process is conducted on a laptop computer with Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz and 8GM of RAM. The programming language is selected for Python 3.8 and compiled on Spyder. The entire training process is using 40,000 data points and resulting in overall 2 hours for 10 epochs.

## III. RESULTS

Training and Testing of the Cross-Entropy Loss MLP classifier and CNN were carried out using a dedicated NVIDIA GeForce 1080 Ti graphics card and 16GM of RAM. The QMI MLP Classifier was trained using a laptop computer with Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz and 8GM of RAM. All the training and testing was on the Fashion-MNIST dataset consisting of 60000 training and 10000 testing samples evenly distributed among 10 classes.

*3.1 Cross-Entropy Loss MLP Classifier*

Training for the MLP over 200 epochs took 5 minutes and 47 seconds to complete. Using the ideal hyperparameters referenced in our methods section this MLP classifier was able to achieve an accuracy of 87.63% on the test data.

| Actual \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 815 | 3 | 23 | 22 | 1 | 1 | 124 | 0 | 11 | 0 |
| 1 | 3 | 971 | 1 | 17 | 6 | 0 | 1 | 0 | 1 | 0 |
| 2 | 17 | 2 | 834 | 11 | 67 | 0 | 68 | 0 | 1 | 0 |
| 3 | 25 | 7 | 15 | 911 | 12 | 0 | 23 | 0 | 7 | 0 |
| 4 | 3 | 2 | 109 | 54 | 781 | 0 | 48 | 0 | 3 | 0 |
| 5 | 1 | 0 | 0 | 1 | 0 | 954 | 0 | 30 | 3 | 11 |
| 6 | 133 | 4 | 94 | 32 | 98 | 0 | 619 | 0 | 20 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 9 | 1 | 966 | 1 | 23 |
| 8 | 6 | 0 | 5 | 7 | 6 | 3 | 4 | 3 | 966 | 0 |
| 9 | 0 | 0 | 0 | 1 | 0 | 11 | 0 | 41 | 1 | 946 |

Table 1. Cross-Entropy Loss MLP Classifier Confusion Matrix on the Fashion-MNIST test dataset.

*3.2 Convolution Neural Network Classifier*

Training for the CNN over 200 epochs took 6 minutes and 35 seconds to complete. Using the ideal hyperparameters referenced in our methods section this classifier obtained an accuracy of 88.4% on the test data.

| Actual \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 836 | 0 | 19 | 23 | 1 | 0 | 110 | 2 | 9 | 0 |
| 1 | 3 | 969 | 2 | 18 | 2 | 0 | 3 | 0 | 3 | 0 |
| 2 | 18 | 0 | 824 | 11 | 60 | 1 | 82 | 0 | 4 | 0 |
| 3 | 25 | 5 | 18 | 900 | 28 | 0 | 24 | 0 | 0 | 0 |
| 4 | 2 | 0 | 126 | 38 | 751 | 0 | 81 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0 | 977 | 0 | 14 | 1 | 8 |
| 6 | 119 | 0 | 84 | 28 | 60 | 0 | 701 | 0 | 8 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 953 | 1 | 31 |
| 8 | 7 | 1 | 2 | 1 | 1 | 5 | 10 | 4 | 969 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 31 | 1 | 960 |

Table 2. CNN Classifier Confusion Matrix on the Fashion-MNIST test dataset.

*3.3 QMI MLP Classifier*

After finishing training, the model is tested on the testing set which has 20,000 data cases. In order to decide which class each test data point belongs to, we design a MAP estimator by using the model output (y) with trained data using label information (C). The MAP estimator is designed in the following way: we assume that each class has a multivariate Gaussian density function. Then for the entire training set, we are maximizing the likelihood, e.g. P(C|y). We didn't set a prior estimation of mean and variance of each multivariate density function because the prior are all the same. The accuracy and testing function is computed by feeding the output of MLP with QMI criterion to MAP estimator. The testing accuracy could achieve 72.06%. The confusion matrix for the testing data is found in Table 3.

| Actual \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1640 | 0 | 39 | 144 | 10 | 15 | 78 | 1 | 27 | 0 |
| 1 | 116 | 1531 | 27 | 258 | 12 | 1 | 5 | 0 | 5 | 0 |
| 2 | 52 | 0 | 1190 | 10 | 371 | 7 | 290 | 0 | 25 | 0 |
| 3 | 164 | 3 | 23 | 1619 | 74 | 6 | 125 | 0 | 5 | 0 |
| 4 | 16 | 0 | 467 | 145 | 1191 | 6 | 170 | 0 | 16 | 0 |
| 5 | 5 | 0 | 3 | 1 | 1 | 1527 | 2 | 388 | 79 | 76 |
| 6 | 513 | 0 | 337 | 60 | 376 | 18 | 580 | 0 | 62 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 220 | 0 | 1706 | 2 | 116 |
| 8 | 19 | 0 | 23 | 78 | 79 | 75 | 46 | 25 | 1626 | 6 |
| 9 | 8 | 0 | 0 | 1 | 2 | 77 | 6 | 167 | 4 | 1758 |

Tabel 3. Confusion matrix of MLP classifier with QMI criterion on the Fashion-MNIST test dataset.

## IV. DISCUSSION

*4.1 Results Comparison*

The inclusion of a SAE to reduce dimensionality and remove noise from the input dataset proved to be hugely beneficial to the classification accuracy of the MLP implementations. Compared to the 85.2% accuracy achieved in [11] the improvement to 87.63% is substantial for classification problems. This improvement brought the accuracy of the MLP more closely comparable to the CNN described in this paper with an accuracy of 88.4% and more competitive with the 91.8% obtained in [11] when taking in account the training time required to reach such accuracies.

With a training time of 5 minutes and 47 seconds the Cross-Entropy MLP was nearly an order of magnitude faster than the CNN in [11] that took approximately 45 minutes to train. However, CNNs still remains more ideal for the image classification problem with the sample space reduced CNN still achieving a greater accuracy than it's MLP counterpart while having only a minor increase in required training time of 48 seconds.

As for QMI, although in our experiments it didn't reach an accuracy that is comparable to a CNN or MLP, we could still see its advantages due to its broad applications. One possible reason that QMI doesn't perform as well as other classifiers is that we didn't train the model with more epochs. Another disadvantage of QMI is that its training process is time consuming compared to other classifiers mentioned in this report. It requires around 2 hours to train for only 10 epochs.

*4.2 Findings*

The stacked autoencoder implemented in this paper proved to be effective at reducing the required input features for classification by a factor of more than 12 while still maintaining a highly accurate reconstruction. This dimensionality change greatly reduced the complexity required of the models to obtain competitive accuracies. As a result the models within this paper could be trained

massively faster than their ground truth training data counterparts.

The encoding of the training data resulted in an improvement to the accuracy of the MLP and a detriment to CNN. This is likely due to the MLP benefiting from the reduced dimensionality and emphasis of certain features allowing the model to better ignore noise within the data and focus on highly relevant individual features. The CNNs reduction in accuracy is likely a result of losing the ability to use 2d convolutions. These convolutions are responsible for recognizing the structural information that is present in the original data that is absent in the encoded data. Following the encoding of the data spatial patterns and relationships are no longer present and it is these connections that allow CNNs to achieve state-of-the-art accuracy on image based classification problems.

Because the QMI loss function gets access to the entire density of the input data, it utilizes much more information than other classifiers. In the future, if we are dealing with big data scenarios, the density estimation will be more accurate which will deliver better results on classification.

## V. CONCLUSION

The Fashion-MNIST is considered a far more difficult problem then both the digit and character MNIST dataset which have been classified at rates exceeding 99%. Based on our experiments, implementing CNN on the original data set would achieve the best performance on the testing set (91.8%).

Even though the QMI criterion is not giving a good performance as we expected, it is still a very powerful approach for classification because it uses the density information of input data. The future work will include prolong the training epoch and do comparison study again, use different kernel size instead of setting it by Silverman's rule, and use QMI criterion in other applications.

Our findings within this paper also continue to add merit to CNNs being more suited to image classification tasks than any other models we have compared. The information garnered from the convolutional operation is incredibly powerful given the simplicity and rather low computation complexity requirement.

The very low reconstruction loss observed from the stacked autoencoder and our different models ability to generalize well on the reduced dimensionality data demonstrates SAE effectiveness. However, while working with SAE it is clear that while great at feature compression, the effective application of autoencoders is very problem, data, and model specific. The large improvement in MLP accuracy and the detrimental effect to the CNN is proof of this.

## VI. REFERENCES


[1] Y. V. Venkatesh, S. Kumar Raja, On the classification of multispectral satellite images using the multilayer perceptron, Pattern Recognition 36 (9) (2003) 2161-2175, https://doi.org/10.1016/S0031-3203(03)00013-X

[2] D. Han, Q. Liu, W. Fan, A new image classification method using CNN transfer learning and web data augmentation, Expert Systems with Applications, 95 (1) (2018) 43-56, https://doi.org/10.1016/j.eswa.2017.11.028

[3] W. Yin, K. Kann, M. Yu, H. Schütze, Comparative Study of CNN and RNN for Natural Language Processing, arXiv:1702.01923 [cs.CL]

[4] J. Park; F. Diehl; M.J.F. Gales; M. Tomalin; P.C. Woodland, Training and adapting MLP features for Arabic speech recognition, 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.

[5] T. Hori, S. Watanabe, Y. Zhang, W. Chan, Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM, arXiv:1706.02737 [cs.CL]

[6] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. 2018. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1063–1072. DOI:https://doi.org/10.1145/3178876.3186005

[7] A. Botalb, M. Moinuddin, U. M. Al-Saggaf and S. S. A. Ali, "Contrasting Convolutional Neural Network (CNN) with Multi-Layer Perceptron (MLP) for Big Data Analysis," 2018 International Conference on Intelligent and Advanced System (ICIAS), Kuala Lumpur, 2018, pp. 1-5, doi: 10.1109/ICIAS.2018.8540626.

[8] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. In Proceedings of the 19th International

Conference on Neural Information Processing Systems (NIPS'06). MIT Press, Cambridge, MA, USA, 153–160.

[9] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[10] Kingma, Diederik P. and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” CoRR abs/1412.6980 (2015): n. pag.

[11] Kruse, Rafe, and Bo Chen. “Analysis and Comparison of MLP and CNN Classification Models.”

[12] Khan, A., Sohail, A., Zahoora, U. and Qureshi, A., 2020. A Survey Of The Recent Architectures Of Deep Convolutional Neural Networks.