Hope you have implemented the paper as-it-is and also developed your own model.

Here are some references for strategies to analyse and improve your model:

1. Practical Aspects of Deep Learning | Deep Learning | WnCC Wiki
2. Loss Functions and Optimization Algorithms | by Apoorva Agrawal | Medium
3. Regularization Techniques | Regularization In Deep Learning
4. Metrics to Evaluate your ML Algorithm | by Aditya Mishra | Towards Data Science
5. Improving Performance of Convolutional Neural Network! | by Dipti Pawar | Medium
6. Ensemble Learning Methods for Deep Learning Neural Networks
7. Note that I had shared links on Hyperparameter Tuning in the last week
8. Utilising Popular CNN Architectures: Researchers have created models and trained them to excel at particular tasks. The popular successfull models with their weights (called as pre-trained models) are available in tensorflow for others to utilise for related tasks. By adding layers to such a model, it can be repurposed to act on a different task. This helps in reducing the training parameters (and thus time) for the new task. This approach is called as Transfer Learning. Pick up a CNN architecture from the above link and either design your own CNN model inspired from it or employ it under transfer learning (for our project task). Compare results with your original.
9. If you have managed to reduce the CNN model training time (remember to use GPU), you can try this out: Split the data into train, validation and test sets. Then, train multiple CNN models under consideration on the train set and evaluate them on the validation set. Choose the one which gave the best validation performance and report its performance on the test set. You can use the K-Fold Cross-Validation technique to reduce the impact of exactly which data got split into train and test set.

Experiment and compare the results and summarize your analysis in a one page document.