Bangk!t JKT3-E Final Project Link

1. Links

* Dataset

<https://www.kaggle.com/minhhuy2810/rice-diseases-image-dataset>

* Improved (Move the Wrong Classified Image) Dataset

<https://drive.google.com/drive/folders/1lbgXCNdc5BICy-MOzRNdfI670TC-BrnL?usp=sharing>

* Paper <https://docs.google.com/document/d/15RwgNITgxPGrTa7jw4eVgjXaWjTC7XSI4GttMkyouZ0/edit?usp=sharing>
* Research notebooks

# [Batch normalization in Neural Networks](https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c)

# [Overfitting and Underfitting With Machine Learning Algorithms](https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/)

# [Deploying Machine learning models on Mobile with Tensorflow Lite and Firebase M.L Kit](https://dev.to/emmarex/deploying-machine-learning-models-on-mobile-with-tensorflow-lite-and-firebase-m-l-kit-4647)

1. [Working with TensorFlow Lite in Flutter](https://itnext.io/working-with-tensorflow-lite-in-flutter-f00d733a09c3)

# [Importing a Keras model into TensorFlow.js](https://www.tensorflow.org/js/tutorials/conversion/import_keras)

1. [Heroku Static File Server](https://github.com/mgonto/heroku-static-file-server)
2. [Cara Deploy Aplikasi React JS ke Heroku](https://medium.com/@ikrisnaw/cara-deploy-aplikasi-react-js-ke-heroku-e1d1b13165c1)

* Github Master Repository

<https://github.com/hansels/bangkit-final-project>

* Github Mobile Application Repository

<https://github.com/hansels/flutter_riceye>

* Github Web Application Repository

<https://github.com/rikeadelia/riceye-webapp>

* Model

<https://drive.google.com/drive/folders/1Xu2wQkN8RXBzSUIaxC6DqA8Zovejo5LY?usp=sharing>

1. Deployed Link (App/web)

* Web : <https://riceye.herokuapp.com>
* App : <https://play.google.com/store/apps/details?id=dev.mtsstudio.riceye>

1. 10 Minute Video recording of your group presentation

* Uploaded to YouTube as unlisted (do not publish public-wide)

<https://www.youtube.com/watch?v=C_c3wUNady8>

1. Link of presentation/slides used in the recording : [Presentation](https://docs.google.com/presentation/d/1uBchdF6SANmeRJm0uFiC_xNkTqE3yL6L2qfctT-6q3g/edit?usp=sharing)

Bangk!t JKT3-E Final Project Paper

*Datasets*

Datasets Link: <https://www.kaggle.com/minhhuy2810/rice-diseases-image-dataset>

Edited Datasets Link: <https://drive.google.com/drive/folders/1lbgXCNdc5BICy-MOzRNdfI670TC-BrnL?usp=sharing>

*Deployment*  
Mobile Application Link: <https://play.google.com/store/apps/details?id=dev.mtsstudio.riceye>  
Website Link: <https://riceye.herokuapp.com/>

*Main Repository*: <https://github.com/hansels/bangkit-final-project>

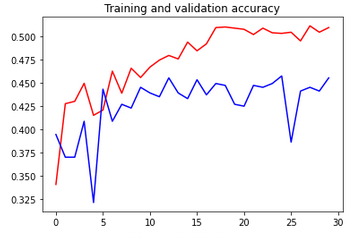
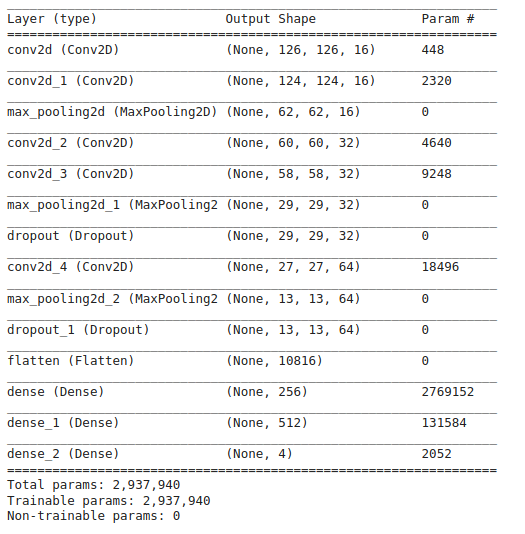
Why do we choose the Rice Diseases Image dataset?

We choose this dataset because this dataset has already a **quite large number** of images for training and testing, and besides that the amount of images are already balanced between all the different classes. The dataset also has **3 types of rice diseases** which are very common diseases in Southeast Asia, such as Healthy, Brown Spot, Hispa and Leaf Blast which are very bad in our case. The dataset also has a **proper work on the image label**, so that we also aren’t required to put some extra work in labeling the images.

In total roughly there are about 3600 images of rice leaf, but 1300 of them are classified as a healthy rice leaf. So to make the data balance, we do the sampling on the healthy rice leaf images to take 500 images. So there are 2,000 data which are divided into 4 classes, which makes 500 images for each class. Every image has 1362x1362 pixels resolution and 3 colors channels (RGB).

Baseline Implementation

We’re taking the same dense layer as the **Convolutional Neural Network course in the Coursera** as our base implementation. The model architecture consists of 2 layers of CNN with 16 kernel size and followed by 2x2 MaxPooling. The output passed to 2 CNNs of 32 kernel size and 2x2 MaxPooling. The last CNN layers stack is single CNN of 64 units and followed by MaxPooling. For classifying, we added 3 stacks of Dense layer with 256, 512, and 4 unit size respectively. Below is the figure of our baseline implementation:



We use the original dataset with Image Augmentation as the same as Coursera. We train this model using Adam optimizer with a learning rate at 0.001. This kind of layer and with some 30 epochs training, lasted pretty quick, for about 90 minutes, but it gave us the testing accuracy of 45%.

Improvement Implementation

From our baseline implementation, as we can see that model accuracy is still not satisfying at all. Although, we choose to improve the **accuracy** of the model. We want to make an improvement in accuracy because the model that we train with Baseline Implementation only gave us 45% of accuracy, it is still very low, we expect the model could give us a higher accuracy and better prediction. So we targeted our model to be accurate at least 85% in the testing accuracy.

So in order to improve the result, we do some things that we think can improve our result. Our improvement are:

1. We try to train in a bigger image size which from 128x128 into 160x160 image size
2. We try to use a Conv-Conv-Pool model architecture
3. We use decay in our Adam Optimizer with decay=initLearningRate/epochs
4. We try to implement an Batch Normalization after each Conv-Conv-Pool layers
5. We implement image augmentation with this setup:

aug = ImageDataGenerator(

rotation\_range=40, width\_shift\_range=0.2,

height\_shift\_range=0.2, shear\_range=0.2,

zoom\_range=0.3,horizontal\_flip=True,

fill\_mode="nearest")

1. We add more layers and more filters in our convolution.

model.add(Conv2D(16, (3, 3), padding="same",input\_shape=inputShape))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(32, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(3, 3)))

model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(64, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3, 3), padding="same"))

model.add(Activation("relu"))

model.add(BatchNormalization(axis=chanDim))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(1024))

model.add(Activation("relu"))

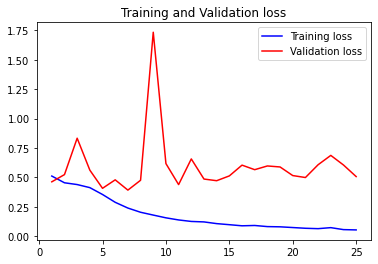
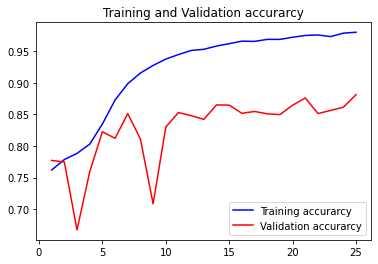
model.add(BatchNormalization())

model.add(Dropout(0.4))

model.add(Dense(n\_classes))

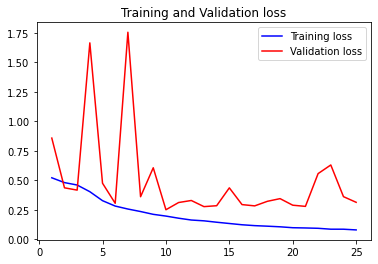
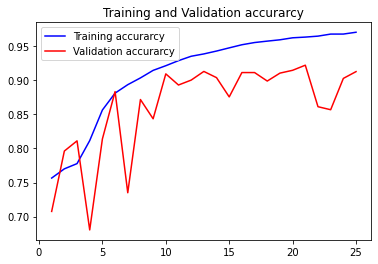
model.add(Activation("softmax"))

Although if we train for another 30 epochs, it’s lasted for about 180 minutes, which is twice much slower than our baseline model training. But the result of the accuracy is 88.1%



Another improvement that we create is, we noticed that some images in the datasets are wrong classified, so we try to fix that by moving the images to the right class. The issue is also discussed in the kaggle discussion. There are roughly about 300 images that we tweak and the result is pretty significant, we managed to improve our accuracy which is from 86% (with the previous dataset) to 91%. The link of the kaggle discussion is

<https://www.kaggle.com/minhhuy2810/rice-diseases-image-dataset/discussion/126486>

These are some image that we move:



Implementation in Indonesia

Indonesia is a country that has a lot of potential from nature. From the beauty of nature for tourism to natural resources that can be produced become a resource for the country's economy and foreign exchange. One of them comes from the agricultural sector. Based on data from katadata.co.id[[1]](#footnote-0), currently, the agricultural sector in Indonesia is the second most influential sector on national economic growth, after the industrial sector.

The agricultural sector is also among the sectors most needed by all Indonesian people, according to data from the Central Statistics Agency (BPS) related to agriculture / agriculture, the number of farmers in Indonesia in 2018 reached 33.48 million people or 12.87% of the total population in Indonesia. Agriculture has an important role for the survival of the people of Indonesia, including all of us. In addition, Indonesia is also a major producer of various tropical agricultural products in the world, making Indonesia one of the main role players in the field of agriculture.

Crop success is determined by many factors. One of them is disease. If the rice plant has disease, then it can be the possibility of crop failure. Farmers can get a loss from 20 until 70 percent due to crop failure[[2]](#footnote-1). Usually, the disease of the rice plant is realized after being spread widely. Farmers oftenly had trouble getting rid of the disease, and resulted in crop failure.

We ideate riceye, a rice detection disease application that uses machine learning. We are making this riceye application to reach out to all of the farmers in Indonesia, therefore we want to publish our application on smartphones.

According to the research[[3]](#footnote-2), smartphone users in Indonesia reach 92 million users in Indonesia only in 2019. This is huge, and an opportunity so we can reach out to more farmers in Indonesia to install the app on their smartphone. They can early detect the condition of the rice plant directly from their smartphone. Farmers can take action quickly after they get the result, rather than wait for the disease to spread widely in their field.

Therefore, with riceye we hope that riceye is a solution to overcome these problems, namely by designing an application whose role is to help detect Rice Disease in farmers Rice Leaf using our knowledge in machine learning.

1. <https://katadata.co.id/berita/2019/05/06/sektor-industri-masih-penyumbang-terbesar-pertumbuhan-ekonomi> [↑](#footnote-ref-0)
2. [Petani Kewalahan Hadapi Hama, Belasan Hektare Terancam Gagal Panen Artikel ini telah tayang di serambinews.com dengan judul Petani Kewalahan Hadapi Hama, Belasan Hektare Terancam Gagal Panen.](https://aceh.tribunnews.com/2020/01/12/petani-kewalahan-hadapi-hama-belasan-hektare-terancam-gagal-panen) [↑](#footnote-ref-1)
3. <https://databoks.katadata.co.id/datapublish/2016/08/08/pengguna-smartphone-di-indonesia-2016-2019> [↑](#footnote-ref-2)