DAFTAR PUSTAKA

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *International Conference on Engineering and Technology (ICET).* Antalya, Turkey: IEEE.

Blake, J. H., Keinath, A. P., & Kluepfel, M. (2018, Desember 13). *Tomato Diseases & Disorders.* Retrieved from Clemson Cooperative Extension: Home & Garden Information Center: https://hgic.clemson.edu/factsheet/tomato-diseases-disorders/

Chen, W., Sun, Q., & Wang, J. (2018). A Novel AdaBoost and CNN Base for Vehicle Classification. *IEEE Access*, 60445-60455.

Garillos-Manliguez. (2016, November 25). Generalized Confusion Matrix for Multiple Classes.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004). *Digital Image Processing using Matlab.* US, Amerika: Pearson Education.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern.* Las Vegas, NV, USA : IEEE.

Kim, T.-H., Park, D.-C., Woo, D.-M., Jeong, T., & Min, S.-Y. (2011). Multi-class Classifier-Based Adaboost Algorithm. *Intelligent Science and Intelligent Data Engineering*, 122-127.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, (pp. 1097-1105). Lake Tahoe, Nevada.

LeChun, Y., Botton, L., Bengio, Y., & Haffiner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 2278 - 2324.

Lee, S.-J., Chen, T., Yu, L., & Lai, C.-H. (2017). Image Classification Based on the oost Convolutional Neural Network. *IEEE Access (Volume: 6)* , 2169-536 .

Rangarajan, A. K., Purushothaman, R., & Ramesh, A. (2018). Tomato crop disease classification using pre-trained deep learning algorithm. *International Conference on Robotics and Smart Manufacturing.* ScienceDirect.

Santosa, B., & Umam, A. (2018). *Data Mining dan Big Data Analytics.* Yogyakarta: Penebar Media Pustaka.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR.*

Singh, V., & Misra, A. (2017). Detection of Plant Leaf Disease using Image Segmentation and Soft Computing Techniques. *Information Processing in Agroculture 4*, 41-49.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2014). Going Deeper with Convolutions.

Vetal, S., & Khule, R. (2017). Tomato Plant Disease Detection usinf Image Processing. *International Journal of Advanced Research in Computer and Communication Engineering*, *6*(6), 293-297.

Yalcin, H., & Razavi, S. (2016). Plant Detection using Convolutional Neural Networks.

Yang, S., Chen, L.-F., Yan, T., Zhao, Y.-H., & Fan, Y.-J. (2017). An Ensemble Classification Algoritm for Convolutional Neural Network based in AdaBoost. *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS).* Wuhan, China : IEEE.

LAMPIRAN

Lampiran 1 Souce Code

CNN 1

```
def cnn_1(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (3, 3), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
    x = Activation("relu")(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(n_classes)(x)
    x = Activation("softmax")(x)
    model = Model(model_input, x, name='cnn_1')
    return model
```

CNN 2

```
def cnn_2(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (5, 5), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
```

CNN 3

```
def cnn_3(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (7, 7), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
    x = Activation("relu")(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(n_classes)(x)
    x = Activation("softmax")(x)
    model = Model(model_input, x, name='cnn_3')
    return model
```

CNN 4

```
def cnn_4(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (3, 3), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (7, 7), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
    ctivation("relu")(x)
    atchNormalization()(x)
    ropout(0.5)(x)
    ense(n_classes)(x)
    ctivation("softmax")(x)
    l = Model(model_input, x, name='cnn_4')
    rn model
```

CNN 5

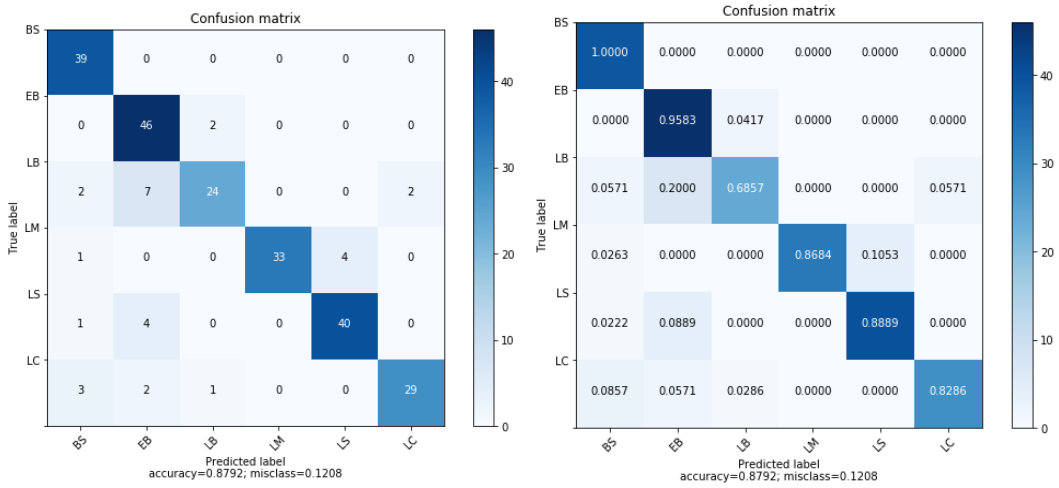
```
def cnn_5(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (3, 3), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
    x = Activation("relu")(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(n_classes)(x)
    x = Activation("softmax")(x)
    model = Model(model_input, x, name='cnn_5')
    return model
```

CNN 6

```
def cnn_6(model_input: Tensor) -> training.Model:
    chanDim = 1
    x = Conv2D(16, (7, 7), padding="same")(model_input)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (5, 5), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Conv2D(16, (3, 3), padding="same")(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = MaxPooling2D(pool_size=(4, 4))(x)
    x = Dropout(0.25)(x)
    x = Flatten()(x)
    x = Dense(512)(x)
    ctivation("relu")(x)
    atchNormalization()(x)
    ropout(0.5)(x)
    ense(n_classes)(x)
    ctivation("softmax")(x)
    l = Model(model_input, x, name='cnn_6')
    rn model
```

Lampiran 2 Performa Model CNN 1

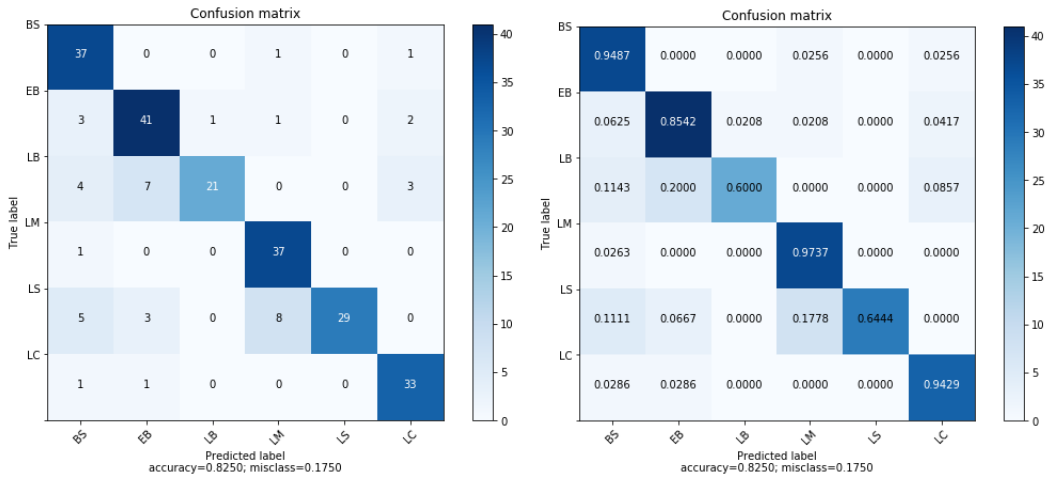
	precision	recall	f1-score	support
BS	0.85	1.00	0.92	39
EB	0.78	0.96	0.86	48
LB	0.89	0.69	0.77	35
LM	1.00	0.87	0.93	38
LS	0.91	0.89	0.90	45
LC	0.94	0.83	0.88	35
micro avg	0.89	0.88	0.88	240
macro avg	0.90	0.87	0.88	240
weighted avg	0.89	0.88	0.88	240
samples avg	0.88	0.88	0.88	240



Gambar 43 *Confusion matrix* evaluasi model CNN 1

Lampiran 3 Performa Model CNN 2

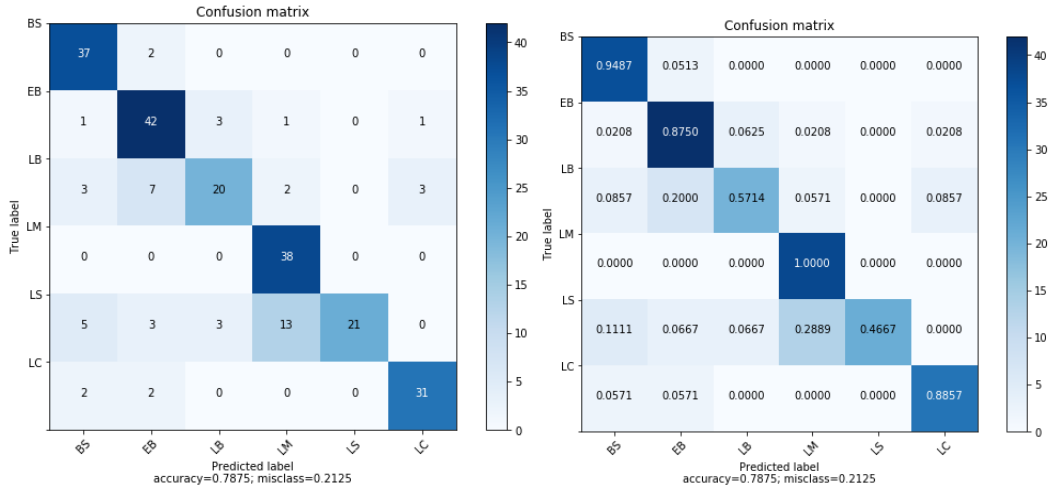
	precision	recall	f1-score	support
BS	0.73	0.95	0.82	39
EB	0.79	0.85	0.82	48
LB	0.95	0.60	0.74	35
LM	0.79	0.97	0.87	38
LS	1.00	0.64	0.78	45
LC	0.85	0.94	0.89	35
micro avg	0.85	0.82	0.84	240
macro avg	0.87	0.83	0.83	240
weighted avg	0.87	0.82	0.83	240
samples avg	0.82	0.82	0.82	240



Gambar 44 *Confusion matrix* evaluasi model CNN 2

Lampiran 4 Performa Model CNN 3

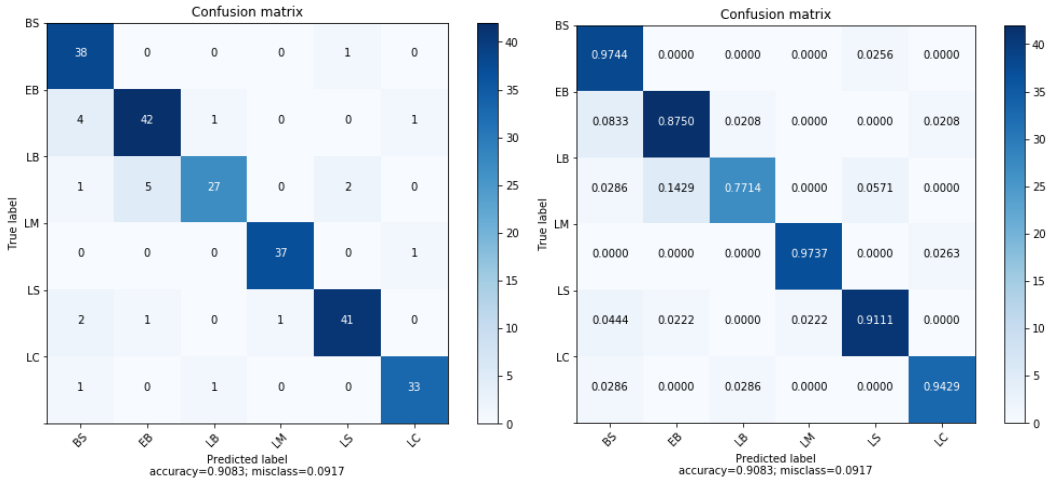
	precision	recall	f1-score	support
BS	0.77	0.95	0.85	39
EB	0.75	0.91	0.82	48
LB	0.83	0.57	0.68	35
LM	0.70	1.00	0.83	38
LS	1.00	0.47	0.64	45
LC	0.89	0.89	0.89	35
micro avg	0.81	0.78	0.80	240
macro avg	0.84	0.79	0.79	240
weighted avg	0.84	0.78	0.79	240
samples avg	0.78	0.78	0.78	240



Gambar 45 *Confusion matrix* evaluasi model CNN 3

Lampiran 5 Performa Model CNN 4

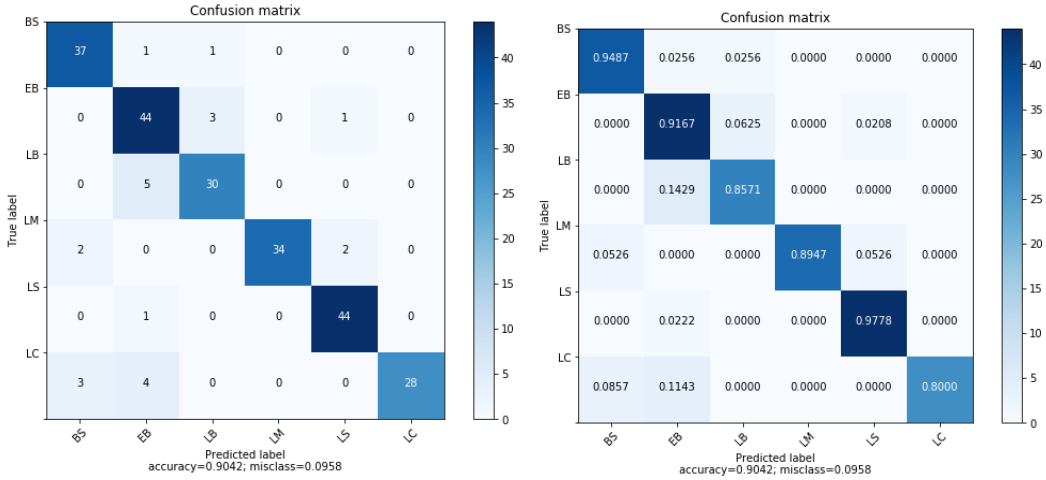
	precision	recall	f1-score	support
BS	0.83	0.97	0.89	39
EB	0.88	0.88	0.88	48
LB	0.93	0.77	0.84	35
LM	0.97	0.97	0.97	38
LS	0.93	0.91	0.92	45
LC	0.94	0.94	0.94	35
micro avg	0.93	0.91	0.92	240
macro avg	0.93	0.91	0.92	240
weighted avg	0.93	0.91	0.92	240
samples avg	0.91	0.91	0.91	240



Gambar 46 *Confusion matrix* evaluasi model CNN 4

Lampiran 6 Performa Model CNN 5

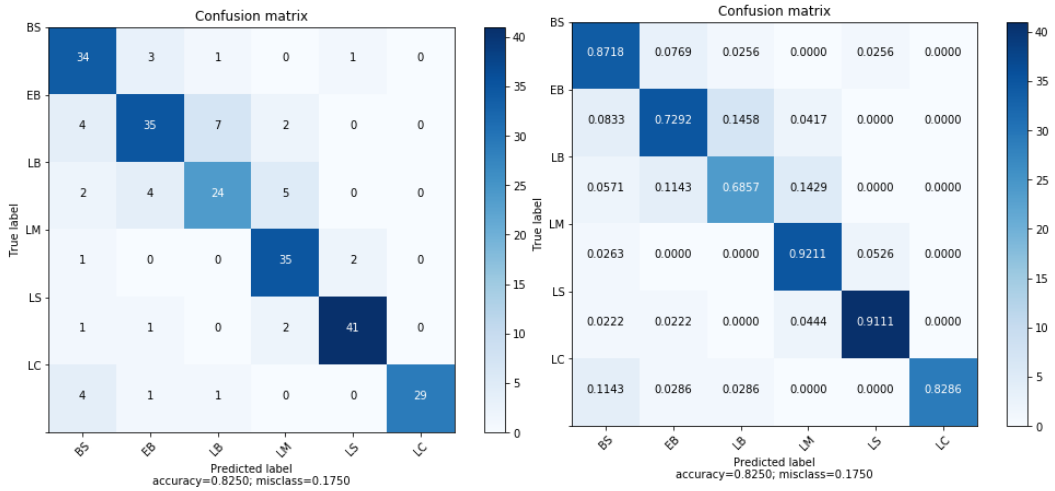
	precision	recall	f1-score	support
BS	0.88	0.95	0.91	39
EB	0.80	0.92	0.85	48
LB	0.88	0.86	0.87	35
LM	1.00	0.89	0.94	38
LS	0.94	0.98	0.96	45
LC	1.00	0.80	0.89	35
micro avg	0.92	0.90	0.91	240
macro avg	0.93	0.90	0.91	240
weighted avg	0.92	0.90	0.91	240
samples avg	0.90	0.90	0.90	240



Gambar 47 *Confusion matrix* evaluasi model CNN 5

Lampiran 7 Performa Model CNN 6

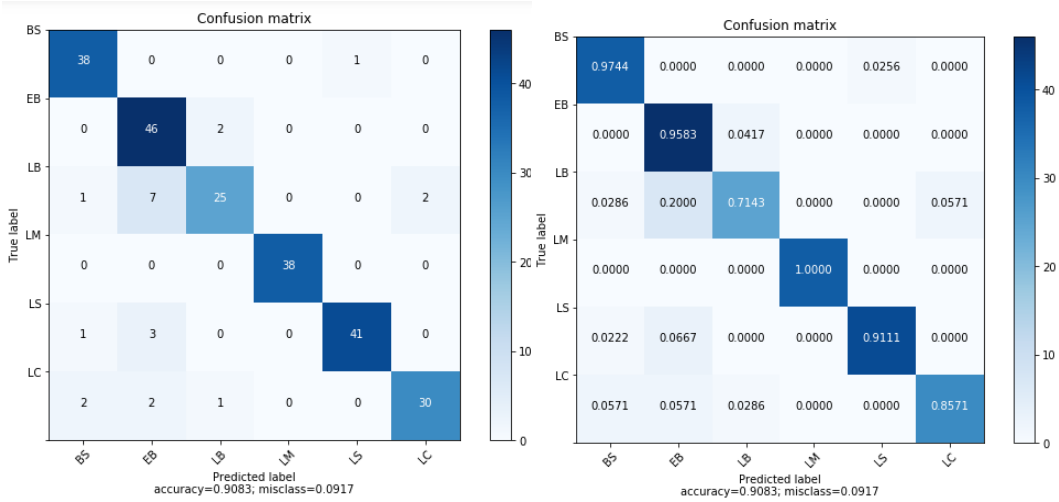
	precision	recall	f1-score	support
BS	0.74	0.87	0.80	39
EB	0.80	0.73	0.76	48
LB	0.73	0.69	0.71	35
LM	0.80	0.92	0.85	38
LS	0.93	0.91	0.92	45
LC	1.00	0.83	0.91	35
micro avg	0.85	0.80	0.83	240
macro avg	0.86	0.80	0.82	240
weighted avg	0.86	0.80	0.83	240
samples avg	0.80	0.80	0.80	240



Gambar 48 *Confusion matrix* evaluasi model CNN 6

Lampiran 8 Performa *Ensemble*

	precision	recall	f1-score	support
BS	0.91	0.97	0.94	39
EB	0.79	0.96	0.87	48
LB	0.89	0.71	0.79	35
LM	1.00	1.00	1.00	38
LS	0.98	0.91	0.94	45
LC	0.94	0.86	0.90	35
micro avg	0.90	0.90	0.90	240
macro avg	0.91	0.90	0.90	240
weighted avg	0.91	0.90	0.90	240
samples avg	0.90	0.90	0.90	240



Gambar 49 *Confusion matrix* evaluasi model *ensemble*

Lampiran 9 Nilai Precision dan Recall data test

Tabel 6 Nilai Precision dan Recall

Kelas	CNN 1		CNN 2		CNN 3		CNN 4		CNN 5		CNN 6		*Ensemble*	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R
BS	0.85	1.00	0.73	0.95	0.77	0.95	0.83	0.97	0.88	0.95	0.74	0.87	0.91	0.97
EB	0.78	0.96	0.79	0.85	0.75	0.91	0.88	0.88	0.80	0.92	0.80	0.73	0.79	0.96
LB	0.89	0.69	0.95	0.60	0.83	0.57	0.93	0.77	0.88	0.86	0.73	0.69	0.89	0.71
LM	1.00	0.87	0.79	0.97	0.70	1.00	0.97	0.97	1.00	0.89	0.80	0.92	1.00	1.00
LS	0.91	0.89	1.00	0.64	1.00	0.47	0.93	0.91	0.94	0.98	0.93	0.91	0.98	0.91
LC	0.94	0.83	0.85	0.94	0.89	0.89	0.94	0.94	1.00	0.80	1.00	0.83	0.94	0.86

Lampiran 10 Akurasi dan waktu komputasi data test

Tabel 7 Akurasi dan waktu komputasi

Model	Akurasi	Waktu Komputasi (detik)
CNN 1	0,8792	2.057665
CNN 2	0,8250	3.999886
CNN 3	0,7875	3.613183
CNN 4	0,9083	3.093516
CNN 5	0,9042	2.324888
CNN 6	0,8250	3.186559
Ensemble	0,9083	9.650408