Monitorowanie trenowania modeli

Weronika Hryniewska

Wywołania zwrotne

- my_callbacks = [
-     tf.keras.callbacks.EarlyStopping(patience=2),
-     tf.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-{val_loss:.2f}.h5'),
-     tf.keras.callbacks.TensorBoard(log_dir='./logs'),
-]
- model.fit(dataset, epochs=10, callbacks=my_callbacks)

Co umożliwia TensorBoard?

https://www.tensorflow.org/tensorboard/get_started

Wiele wejść, wiele wyjść i zespoły klasyfikatorów

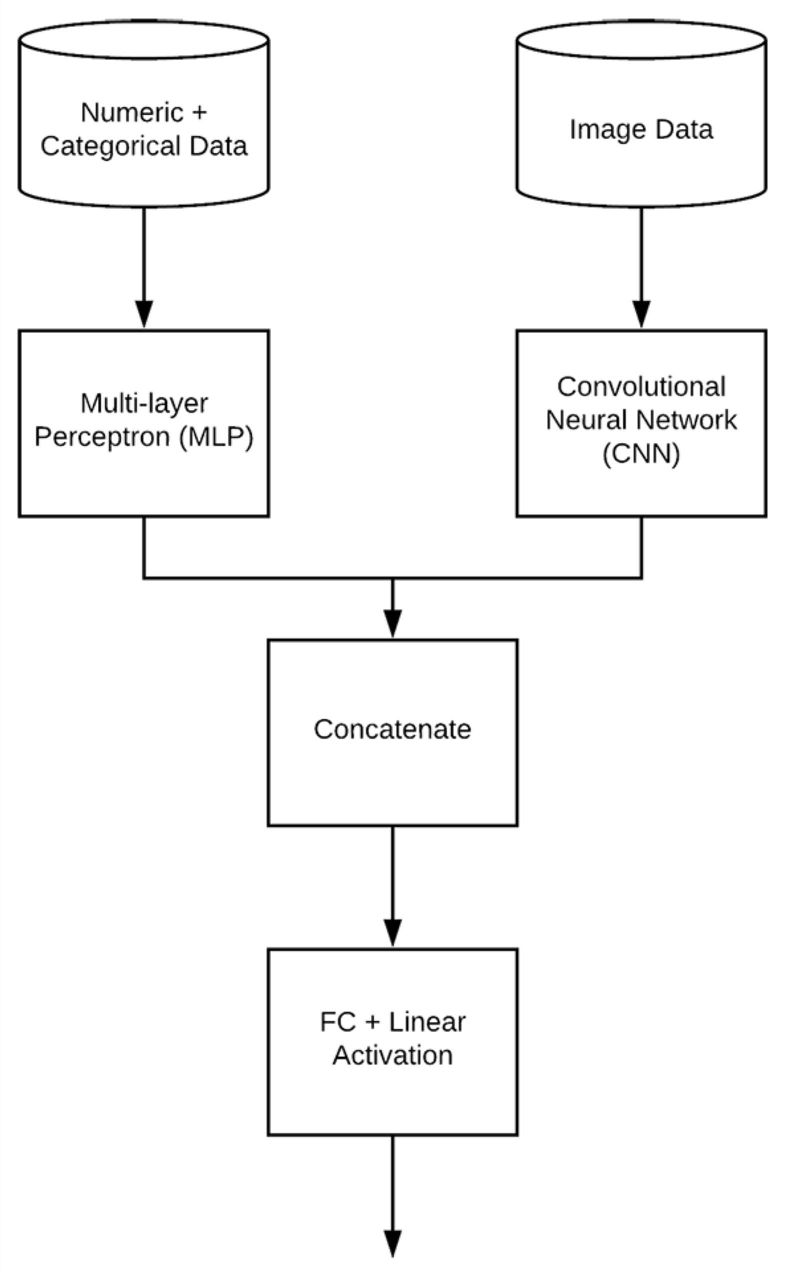
Weronika Hryniewska

Dwa sposoby definiowania modelu

```
model = Sequential()
model.add(Dense(8, input_shape=(10,), activation="relu"))
model.add(Dense(4, activation="relu"))
model.add(Dense(1, activation="linear")
```

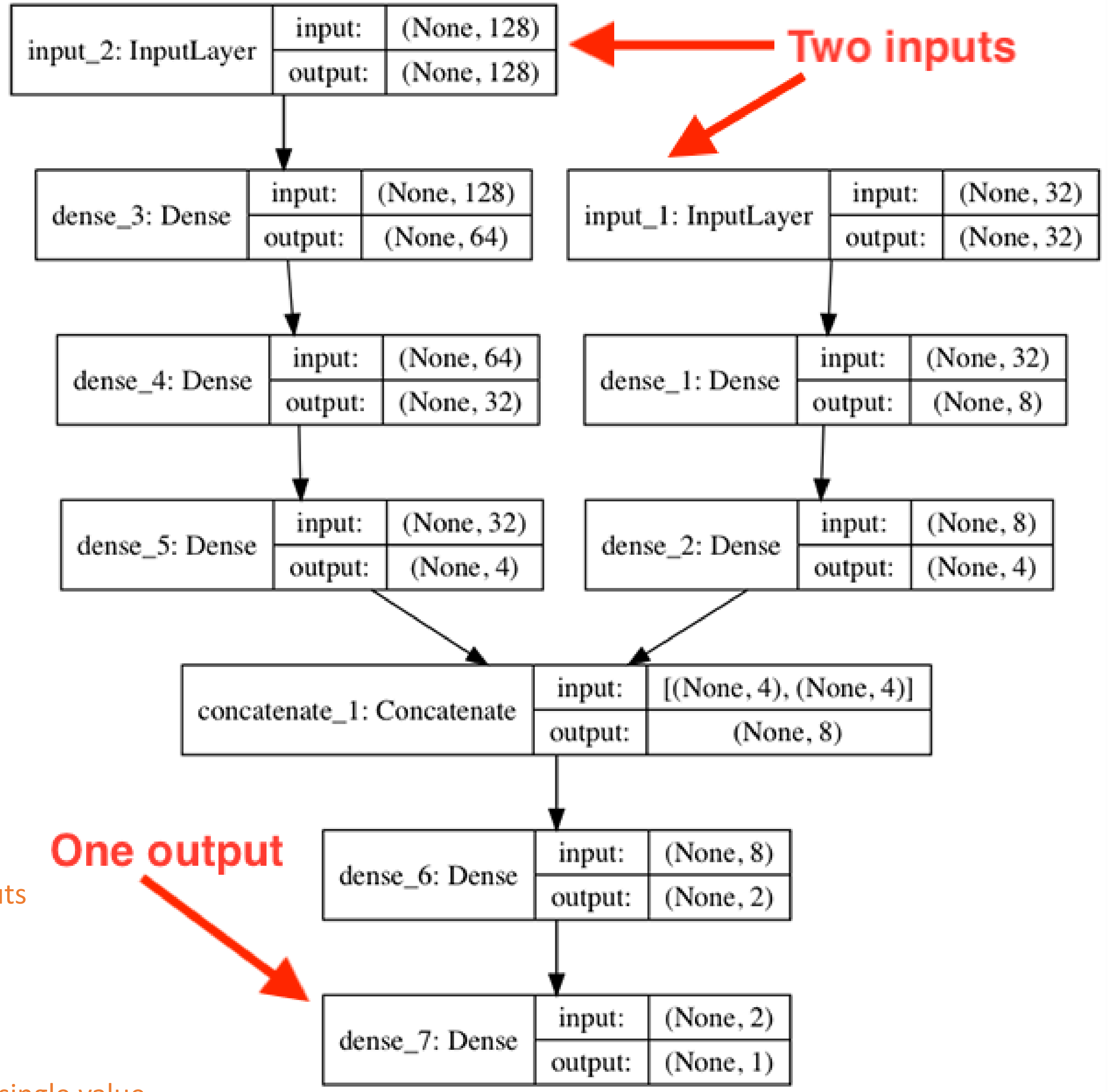
```
inputs = Input(shape=(10,))
x = Dense(8, activation="relu")(inputs)
x = Dense(4, activation="relu")(x)
x = Dense(1, activation="linear")(x)
model = Model(inputs, x)
```

Dwa wejścia


Numeric + Categorical Data
Image Data
Multi-layer Perceptron (MLP)
Convolutional Neural Network (CNN)
Concatenate
FC + Linear Activation

www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data

```
# define two sets of inputs
inputA = Input(shape=(32,))
inputB = Input(shape=(128,))

# the first branch operates on the first input
x = Dense(8, activation="relu")(inputA)
x = Dense(4, activation="relu")(x)
x = Model(inputs=inputA, outputs=x)

# the second branch opreates on the second input
y = Dense(64, activation="relu")(inputB)
y = Dense(32, activation="relu")(y)
y = Dense(4, activation="relu")(y)
y = Model(inputs=inputB, outputs=y)

# combine the output of the two branches
combined = concatenate([x.output, y.output])

# apply a FC layer and then a regression prediction on the combined outputs
z = Dense(2, activation="relu")(combined)
z = Dense(1, activation="linear")(z)

# our model will accept the inputs of the two branches and then output a single value
model = Model(inputs=[x.input, y.input], outputs=z)
```

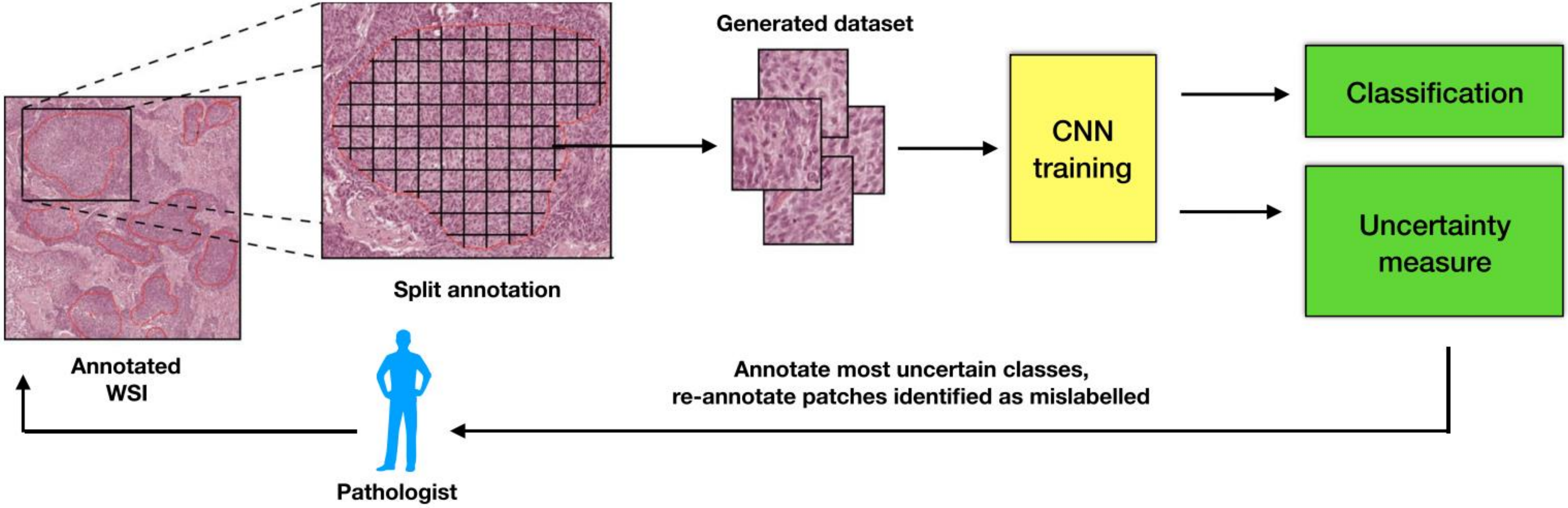

input_2: InputLayer | input: (None, 128) | output: (None, 128)
dense_3: Dense | input: (None, 128) | output: (None, 64)
input_1: InputLayer | input: (None, 32) | output: (None, 32)
dense_4: Dense | input: (None, 64) | output: (None, 32)
dense_1: Dense | input: (None, 32) | output: (None, 8)
dense_5: Dense | input: (None, 32) | output: (None, 4)
dense_2: Dense | input: (None, 8) | output: (None, 4)
concatenate_1: Concatenate | input: [(None, 4), (None, 4)] | output: (None, 8)
dense_6: Dense | input: (None, 8) | output: (None, 2)
dense_7: Dense | input: (None, 2) | output: (None, 1)

Two inputs

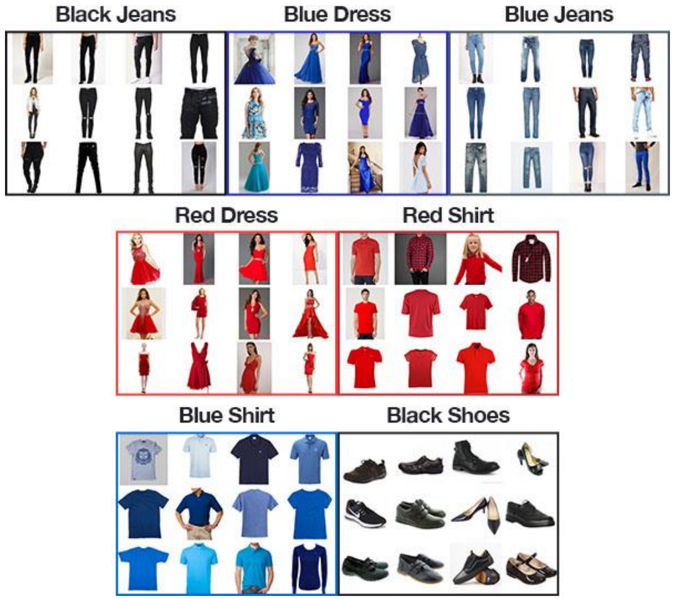
One output

www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data

Dwa wyjścia



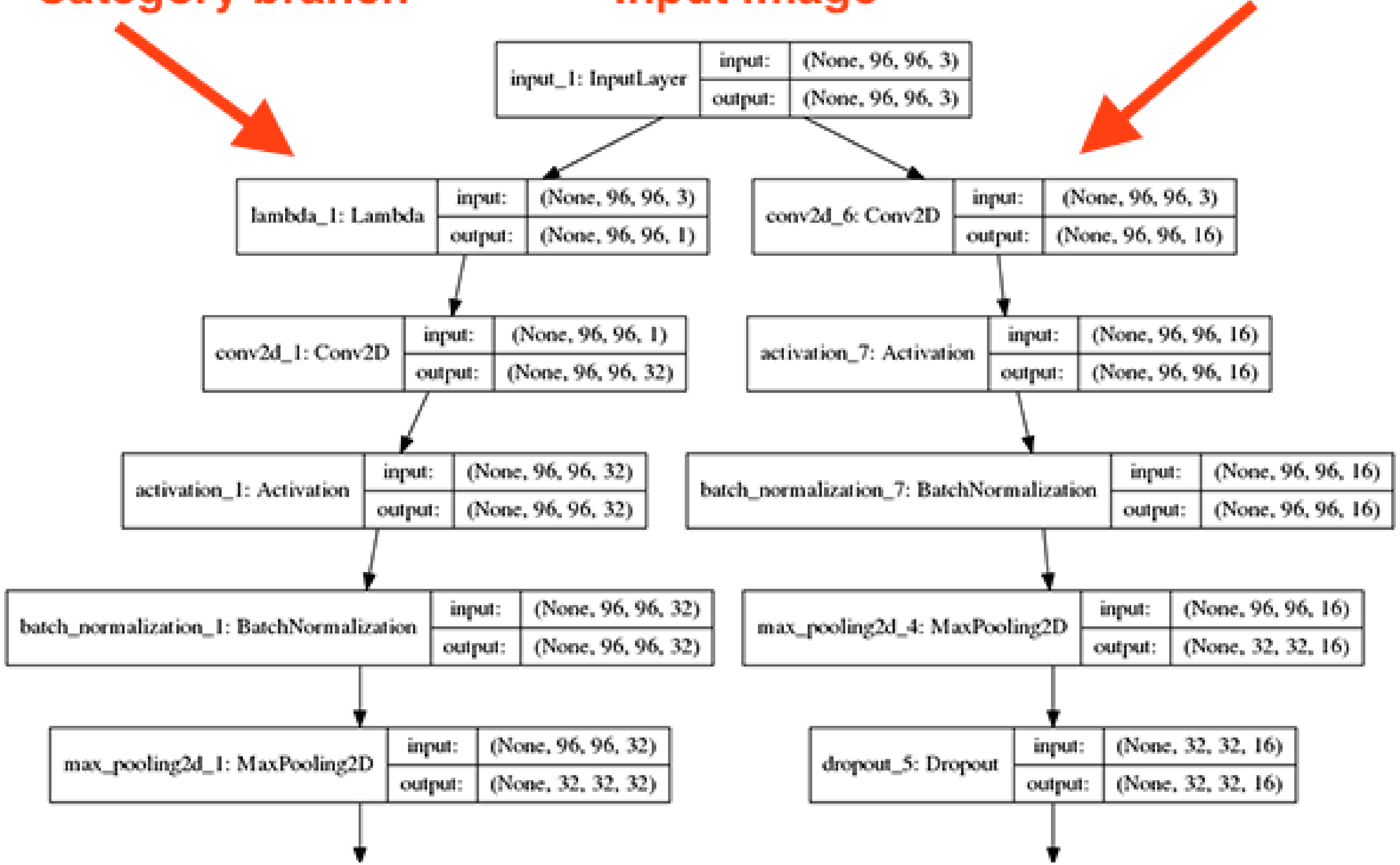
Rączkowski, Ł., Możejko, M., Zambonelli, J. *et al.* ARA: accurate, reliable and active histopathological image classification framework with Bayesian deep learning. *Sci Rep* **9,** 14347 (2019). https://doi.org/10.1038/s41598-019-50587-1



Category branch

Input image

Color branch



input_1: InputLayer | input: (None, 96, 96, 3) | output: (None, 96, 96, 3)

lambda_1: Lambda | input: (None, 96, 96, 3) | output: (None, 96, 96, 1)

conv2d_1: Conv2D | input: (None, 96, 96, 1) | output: (None, 96, 96, 32)

activation_1: Activation | input: (None, 96, 96, 32) | output: (None, 96, 96, 32)

batch_normalization_1: BatchNormalization | input: (None, 96, 96, 32) | output: (None, 96, 96, 32)

max_pooling2d_1: MaxPooling2D | input: (None, 96, 96, 32) | output: (None, 32, 32, 32)

conv2d_6: Conv2D | input: (None, 96, 96, 3) | output: (None, 96, 96, 16)

activation_7: Activation | input: (None, 96, 96, 16) | output: (None, 96, 96, 16)

batch_normalization_7: BatchNormalization | input: (None, 96, 96, 16) | output: (None, 96, 96, 16)

max_pooling2d_4: MaxPooling2D | input: (None, 96, 96, 16) | output: (None, 32, 32, 16)

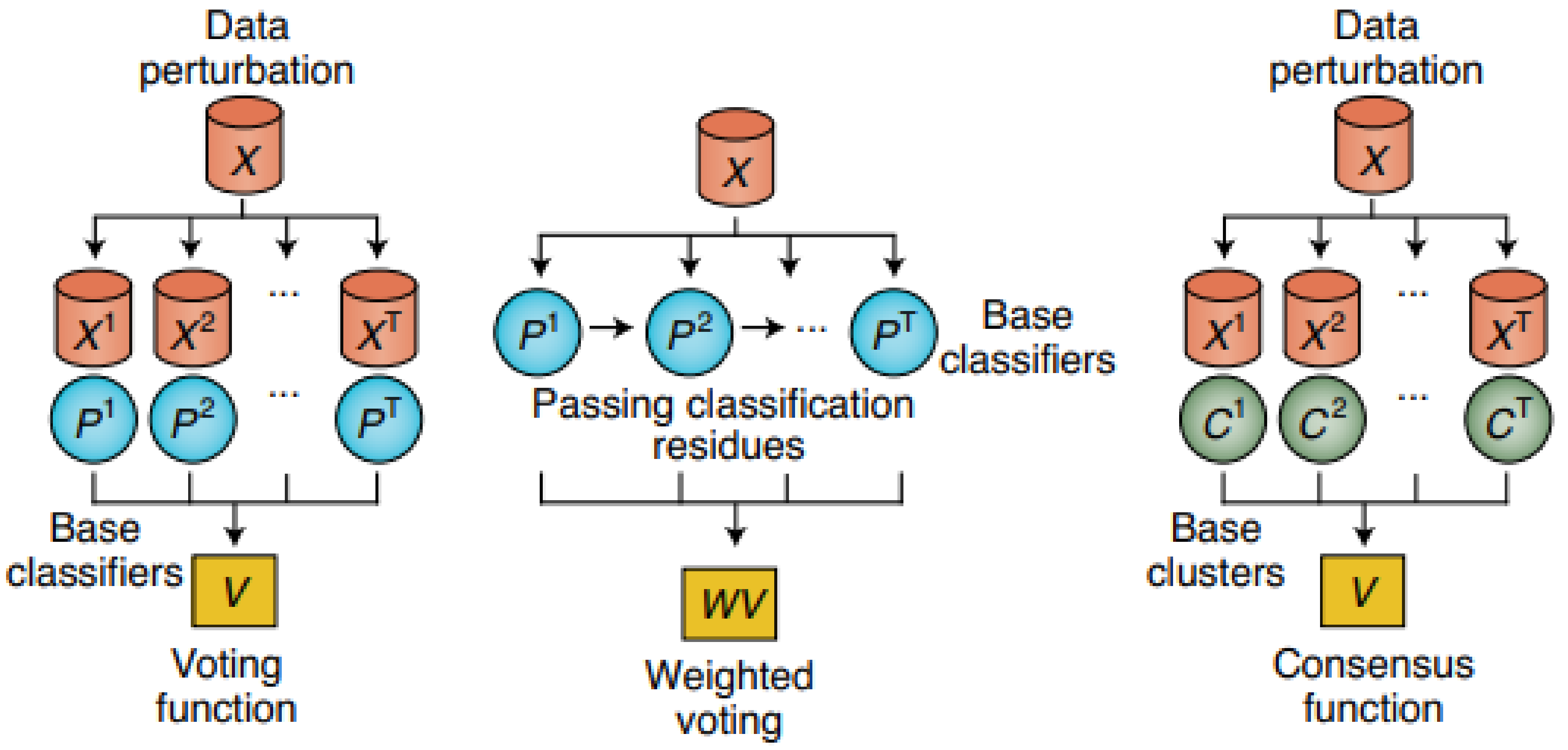
dropout_5: Dropout | input: (None, 32, 32, 16) | output: (None, 32, 32, 16)

www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses

```
class FashionNet:
        @staticmethod
        def build_category_branch(inputs, numCategories, finalAct="softmax", chanDim=-1):
                …
                x = Dense(numCategories)(x)
                x = Activation(finalAct, name="category_output")(x)
                return x


losses = {"category_output": "categorical_crossentropy",  "color_output":
    "categorical_crossentropy"}
lossWeights = {"category_output": 1.0, "color_output": 1.0}

model.compile(optimizer=Adam(), loss=losses, loss_weights=lossWeights,
metrics=["accuracy"])
```

www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses

Zespoły modeli



bagging-, boosting-, stacking-based methods

Cao, Y., Geddes, T.A., Yang, J.Y.H. *et al.* Ensemble deep learning in bioinformatics. *Nat Mach Intell* **2,** 500–508 (2020). https://doi.org/10.1038/s42256-020-0217-y