**Dropconnect**

Dropconnect is een generalisatie van de Dropout layer. Bij Dropout wordt een random geselecteerde subset van activaties naar 0 gezet in elke laag. Bij Dropconnect wordt een random geselecteerde subset van weights op nul gezet.

Dropout helpt bij het verminderen van over-fitting en verbetert de performance op de test set.

Om de weight matrix in een DropConnect layer te updaten wordt een mask toegepast op de gradient zodat alleen de elementen die actief waren bij de forward propagation worden geüpdate.

Zowel dropout als dropconnect geven optimale waarden bij een drop rate in de buurt van 0.5. Zonder gebruik van Dropout of DropConnect overfit het model snel, terwijl bij gebruik van deze methoden de convergence langzamer is, maar de test error uiteindelijk ook lager.

DropConnect helpt bij het generaliseren van grote neural networks. DropConnect doet het vaak beter dan Dropout. DropConnect zorgt ervoor dat je grote modellen kan trainen zonder overfitting.

<http://proceedings.mlr.press/v28/wan13.pdf>

Bij dropout zet je de waarde van enkele neurons op 0 Er is een bepaalde kans p (bv. 0.5) dat de waarde van een neuron op 0 wordt gezet. Dit zorgt ervoor dat er bij een specifieke trainingsstap niets geleerd wordt, maar het netwerk moet leren gebruikmakende van de andere neurons.

Bij DropConnect zet je geen neurons op 0, maar de gewichten tussen neurons. Je hebt dus een bepaalde kans dat verbindingen tussen nodes worden weggehaald. Dit maakt deze methode algemener dan Dropout.

Convolutional neural networks doen het vaak niet goed met dropout layers, doordat sommige filters cruciaal kunnen zijn voor de final output. Dropout wordt tegenwoordig minder gebruikt met CNNs, omdat batch normalization deze techniek grotendeels heeft vervangen. Je standaardiseerd hierbij de output van verschillende layers.

[Preventing Overfitting in Neural Networks with Dropout and DropConnect! - YouTube](https://www.youtube.com/watch?v=atzUm4mrLn8)

(Sorry voor het enthousiasme van deze guy)

Er is helaas geen DropConnect layer beschikbaar in tensorflow.keras

[Module: tf.keras.layers  |  TensorFlow Core v2.8.0](https://www.tensorflow.org/api_docs/python/tf/keras/layers)

Hoewel er geen DropConnect layer is, staat hier op Github wellicht wel code waarmee DropConnect kan worden gebruikt: [teelinsan/KerasDropconnect: An implementation of DropConnect Layer in Keras (github.com)](https://github.com/teelinsan/KerasDropconnect).

Discussie op stackoverflow over dropconnect layer in Keras: [python - Is there a DropConnect layer in Keras? - Stack Overflow](https://stackoverflow.com/questions/44199844/is-there-a-dropconnect-layer-in-keras)

**Batch normalization**

Normalisatie en standaardisatie transformeren de data zodat alles van dezelfde ordegrootte is. Grootte waarden in niet genormaliseerde datasets kunnen voor instabiliteit zorgen in een netwerk. Het zorgt voor het exploderen van gradiënten. Het trainen van het netwerk gaat hierdoor veel minder goed/snel.

Batch normalization normaliseert de output van de activatie functie. Vervolgens wordt de output vermenigvuldigd met een willekeurige parameter en telt er een willekeurige bij op. Deze parameters zijn ook trainbaar. Batch normalization kan per laag worden toegepast. Het vindt plaats per batch en hangt dus af van de batch size die je gekozen hebt. Batch normalization is beschikbaar in Keras!

[Batch Normalization (“batch norm”) explained - YouTube](https://www.youtube.com/watch?v=dXB-KQYkzNU)

Batch normalization zorgt ervoor dat de mean output en de variance van de hidden layers constant blijven. De eerdere lagen veranderen minder snel gezien vanuit diepere lagen. Hierdoor kunnen diepere lagen makkelijker leren.

[Why Does Batch Norm Work? (C2W3L06) - YouTube](https://www.youtube.com/watch?v=nUUqwaxLnWs)