Regularization in Deep Learning

`from keras.` `regularizers` `import` `l1, l2`

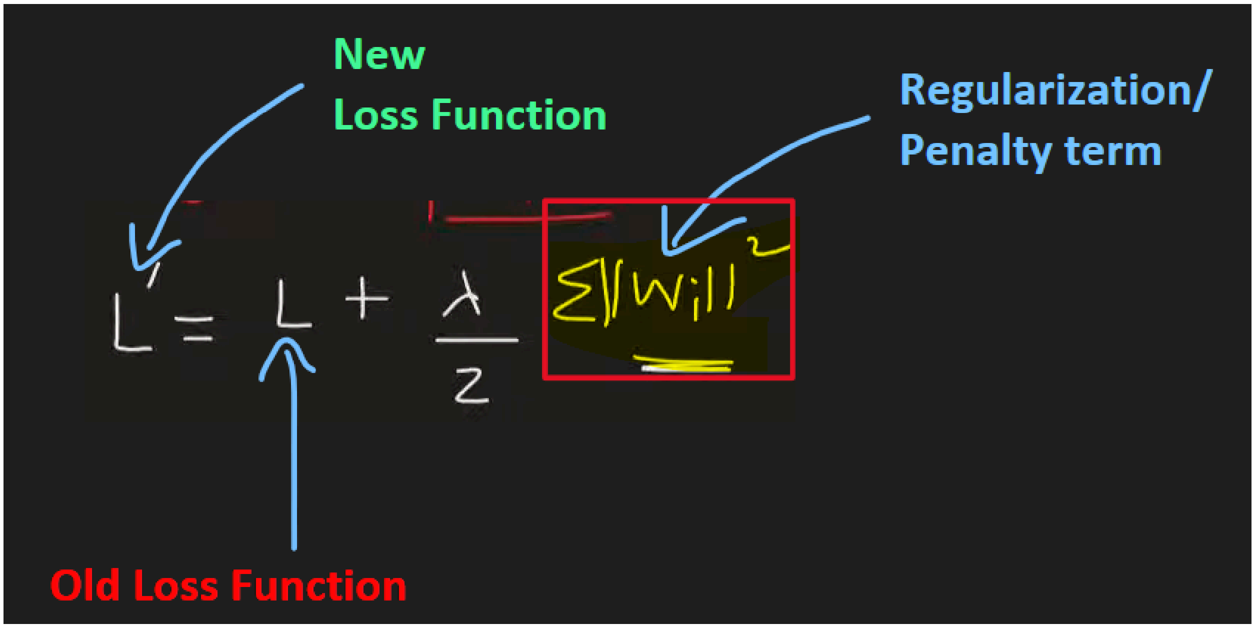
`kernel_regularizer` `=` `l2(0.01)`

- Regularization techniques help neural networks **generalize better** by **reducing overfitting**.
- Overfitting occurs when a model learns not only the underlying patterns in the data but also the noise or irrelevant details, which leads to poor generalization to new, unseen data.

Regularization helps by:

- Keeping **weights small**🔽
- Avoiding **too complex** decision boundaries
- Improving **generalization**

- Regularization adds a factor → w_i^2 (For L2 regularization)
- It reduces the overall weight



🚀 Types of Regularization

L1 & L2 Regularization (Weight Decay)

- **L1 (Lasso)**: Adds `λ|w|` to loss → encourages **sparsity** (**some weights = 0**).
- **L2 (Ridge)**: Adds `λw²` to loss → shrinks weights smoothly.

```
from keras.regularizers import l1, l2

model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))  # L2
model.add(Dense(64, activation='relu', kernel_regularizer=l1(0.01)))  # L1
```

✅ 1. L1 Regularization (Lasso)

- Adds **sum of absolute weights** to loss function:

$$\text{Loss} = \text{Original Loss} + \lambda \sum |w_i|$$

- w_i are the model's weights.
- λ is the regularization strength (also known as the regularization coefficient).

- Encourages **sparsity** (many weights become 0)
- Helps with **feature selection**

✅ 2. L2 Regularization (Ridge) (IMP)

- Adds **sum of squared weights** to loss:

$$\text{Loss} = \text{Original Loss} + \lambda \sum w_i^2$$

- Penalizes large weights, but doesn't force them to zero
- Encourages **weight smoothing**

> **You don't do this for biases.**
>
> $$\cancel{\sum_{i=1}^{K} \|b_i\|^2}$$

💡 Tip:

- **L2 regularization is generally more common than L1** because it doesn't produce sparse weights (i.e., most weights will remain non-zero).
- **L1 regularization** can lead to sparse models with some weights being exactly zero, which can be useful for feature selection.

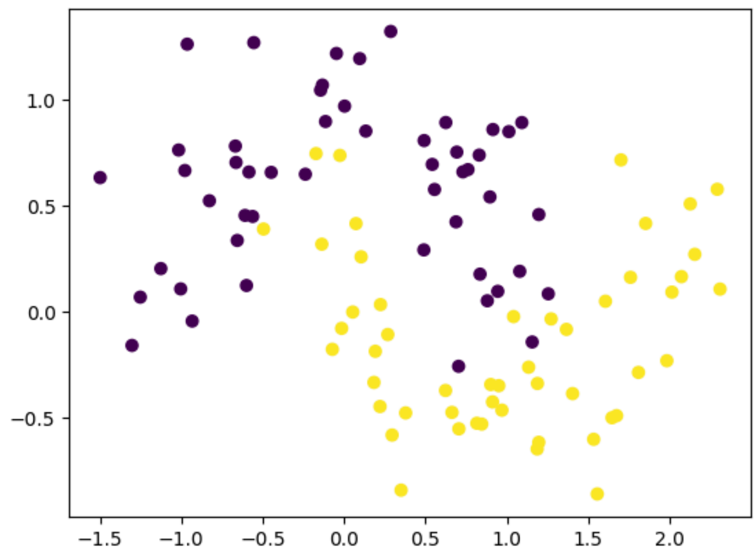
💡 **🔼λ → Strong Regularization (towards underfitting)**

🔽λ → Weak Regularization

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
import seaborn as sns
from mlxtend.plotting import plot_decision_regions

import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
```

```
X, y = make_moons(100, noise=0.25,random_state=2)
import matplotlib.pyplot as plt
plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
```



Without regularization:

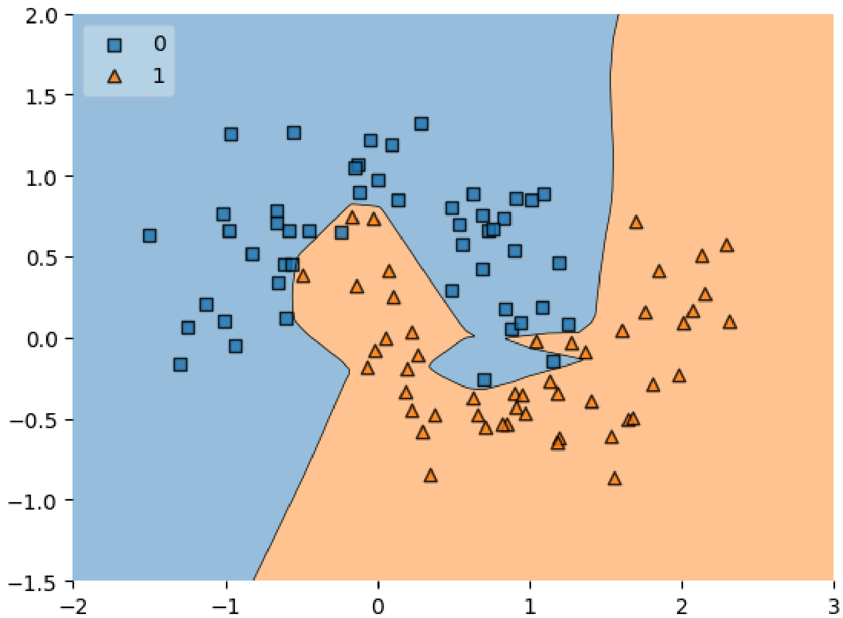
```
model1 = Sequential()

model1.add(Dense(128,input_dim=2, activation="relu"))
model1.add(Dense(128, activation="relu"))
model1.add(Dense(1,activation='sigmoid'))

model1.summary()
```

```
Model: "sequential"

┌─────────────────────────────┬──────────────────────────┬─────────────┐
│ Layer (type)                │ Output Shape             │     Param # │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ dense (Dense)               │ (None, 128)              │         384 │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ dense_1 (Dense)             │ (None, 128)              │      16,512 │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ dense_2 (Dense)             │ (None, 1)                │         129 │
└─────────────────────────────┴──────────────────────────┴─────────────┘

 Total params: 17,025 (66.50 KB)

 Trainable params: 17,025 (66.50 KB)

 Non-trainable params: 0 (0.00 B)
```

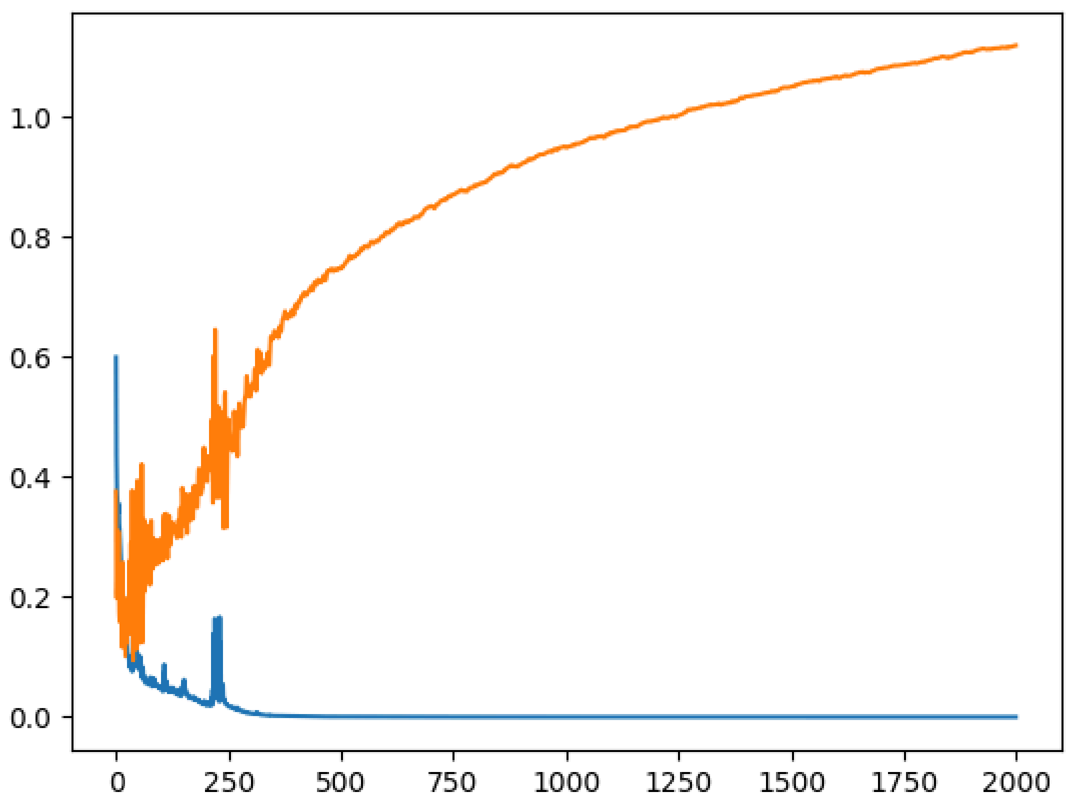
```
adam = Adam(learning_rate=0.01)
model1.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])

history1 = model1.fit(X, y, epochs=2000, validation_split = 0.2,verbose=0)
```

👆 Took **3 mins, 6 sec** to run

```
plot_decision_regions(X, y.astype('int'), clf=model1, legend=2)
plt.xlim(-2,3)
plt.ylim(-1.5,2)
plt.show()
```




```
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
```



With Regularization:

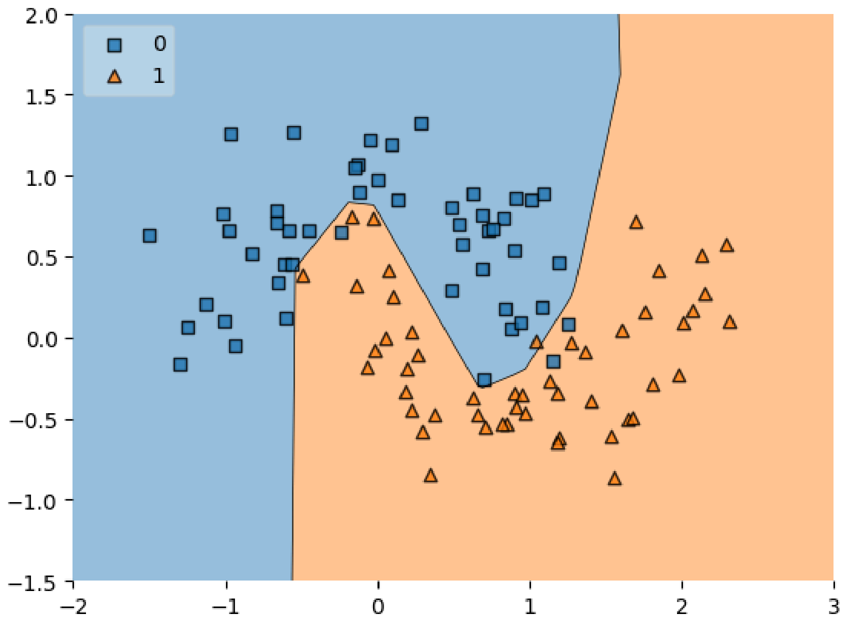
```
model2 = Sequential()

model2.add(Dense(128,input_dim=2, activation="relu",kernel_regularizer=tens
orflow.keras.regularizers.l1(0.001)))
model2.add(Dense(128, activation="relu",kernel_regularizer=tensorflow.keras.
regularizers.l1(0.001)))
model2.add(Dense(1,activation='sigmoid'))
```

💡 **The <u>default</u> values used are `l1=0.01` and `l2=0.01`**

```
adam = Adam(learning_rate=0.01)
model2.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accur
acy'])

history2 = model2.fit(X, y, epochs=2000, validation_split = 0.2,verbose=0)
```

👆 ***3 Mins*** to run

```
plot_decision_regions(X, y.astype('int'), clf=model2, legend=2)
plt.xlim(-2,3)
plt.ylim(-1.5,2)
plt.show()
```




```
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
```