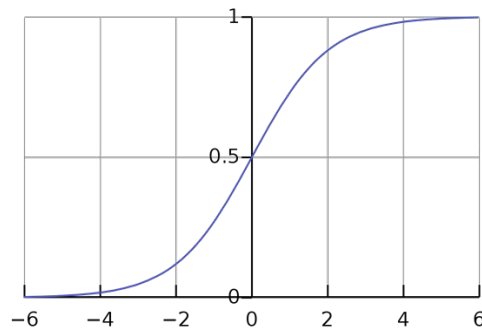
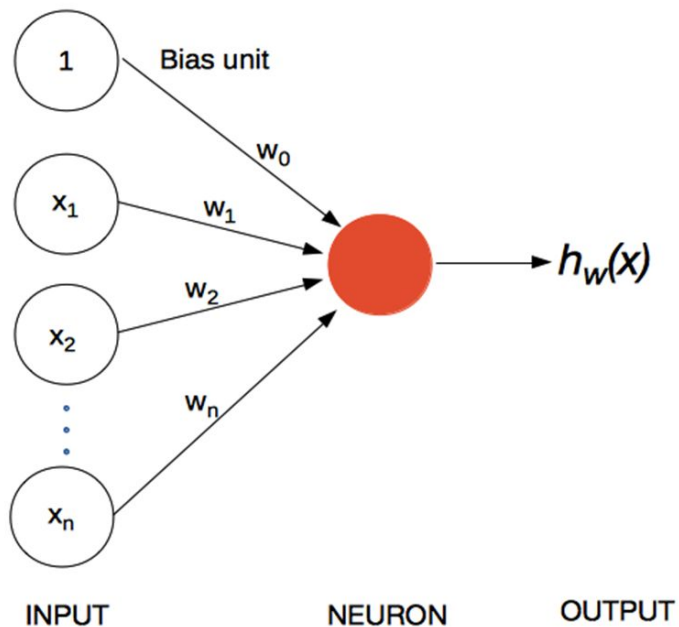


# KNSI Golem Bootcamp

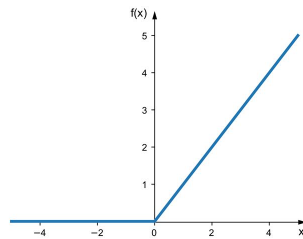
## Level Up

Spotkanie 0 - Keras

# Previously on Bootcamp...



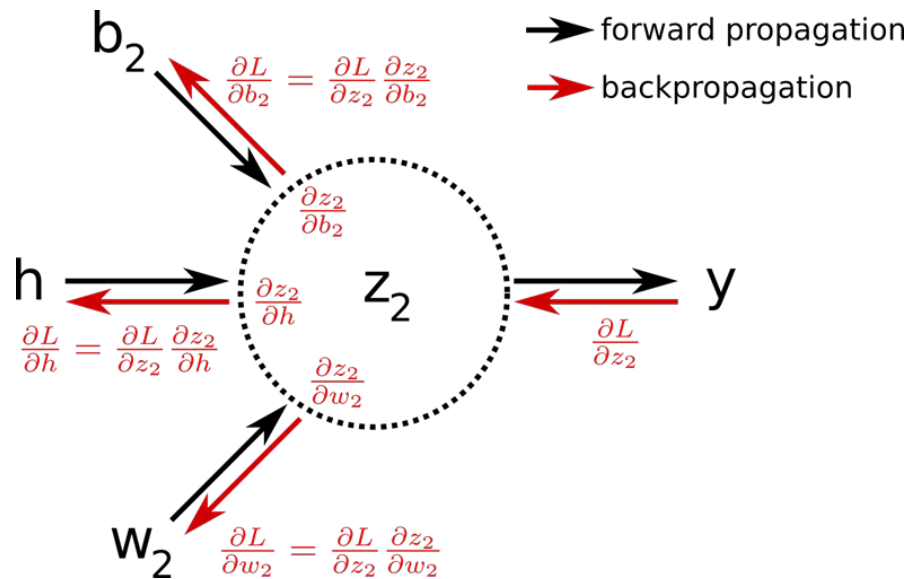
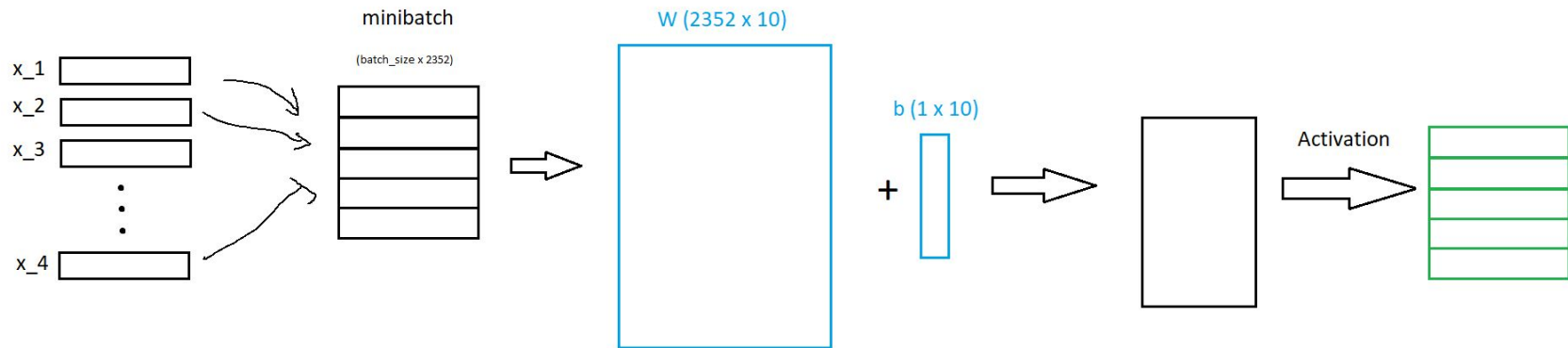
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



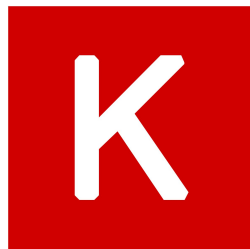
$$ReLU(z) = \max(0, z)$$

# Previously on Bootcamp...

- Forward pass
- Backward pass

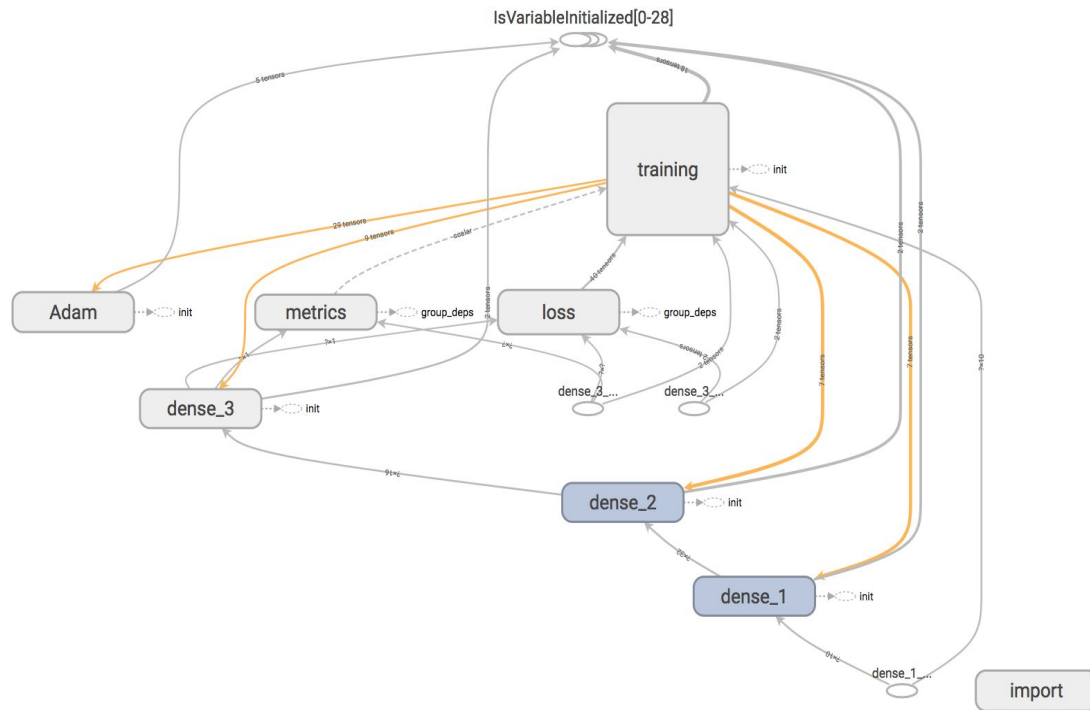


# Biblioteki do Deep Learning



# Biblioteki do Deep Learning, c.d.

1. Automatyczne tworzenie Grafu Obliczeń
2. Automatyczne obliczanie gradientu (Backpropagation)
3. Obliczenia na GPU



# Keras. Zalety

1. Prościej się w nim pisze
2. Niektóre rzeczy podobne do Sklearn (np. metoda *.fit()* )
3. Większość API i koncepcji się przeplata w innych frameworkach

# Spotkanie 0. Co nas czeka?

- Tensory, operacje na nich
- Warstwy
- Klasyfikacja vs. Regresja w Sieciach Neuronowych
- Warstwa Softmax
- Maximum Likelihood Principle
- Co robić z przeuczeniem?
- Regularyzacja (L1, L2, Dropout)
- Batch Normalization
- Inicjowanie wag
- PRAKTYKA



# Tensory

- Podstawowa struktura danych w sieciach neuronowych

A tensor is an N-dimensional array of data



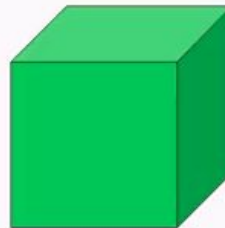
Rank 0  
Tensor  
scalar



Rank 1  
Tensor  
vector



Rank 2  
Tensor  
matrix



Rank 3  
Tensor



Rank 4  
Tensor



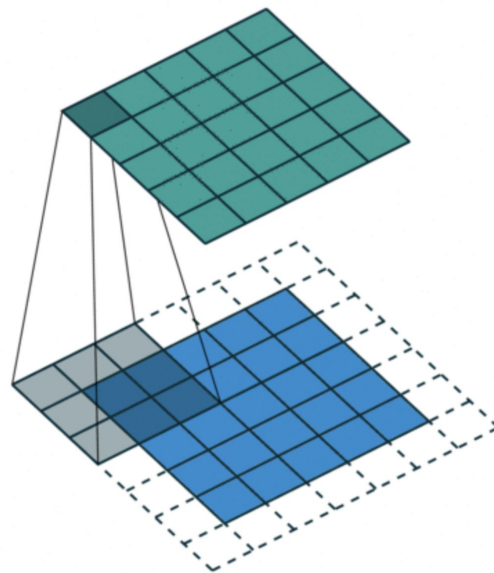
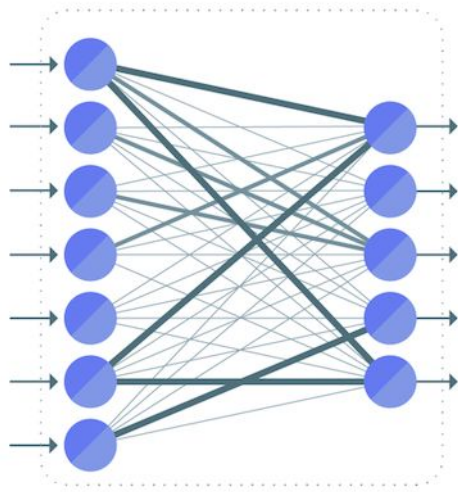
# Obliczenia na GPU

- O wiele wydajniej jest zwektoryzować obliczenia
- Pojedyncze ładowanie tensora do pamięci (nie tracimy czas na dostęp do dysku)

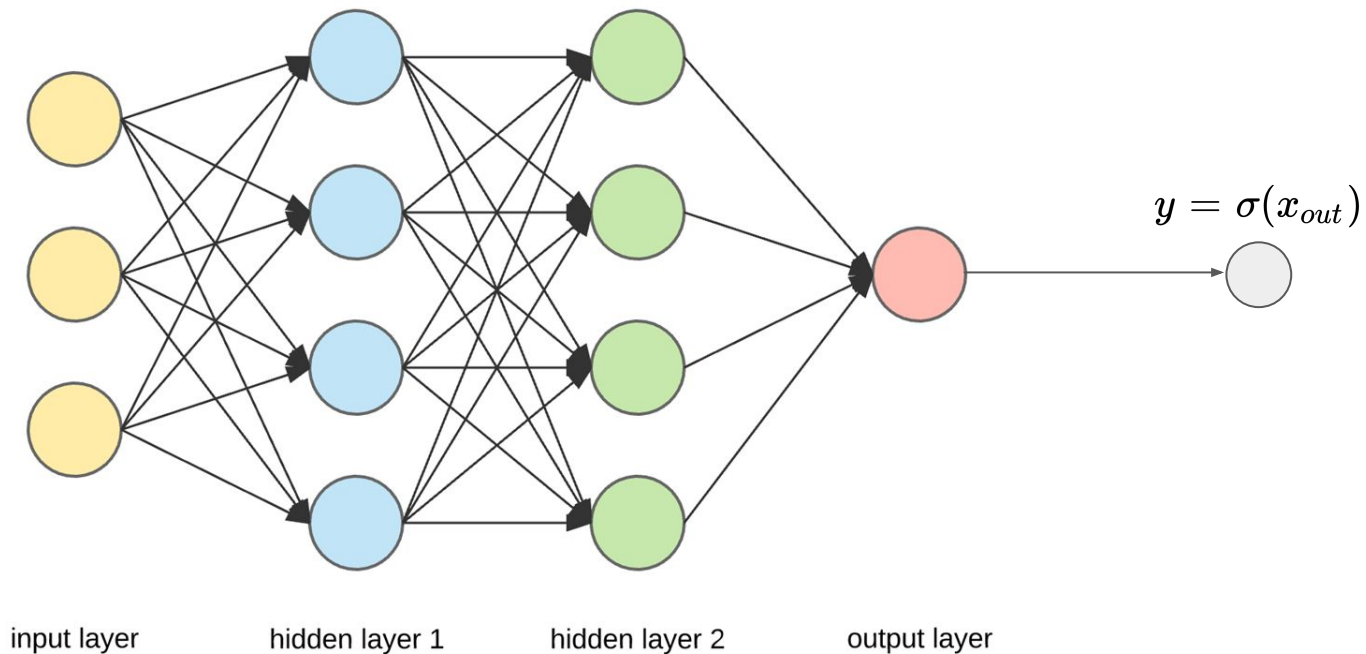
```
import keras
import tensorflow as tf
config = tf.ConfigProto( device_count = {'GPU': 1 , 'CPU': 56} )
sess = tf.Session(config=config)
keras.backend.set_session(sess)
```



# Warstwy



# Przypomnienie: Architektura Fully Connected



# Klasyfikacja vs. Regresja w Sieciach Neuronowych

## Regresja

Ostatnia warstwa sieci:

- To, ile wartości przewidujemy

Funkcja kosztu:

- MSE / RMSE / MAE

## Klasyfikacja

Ostatnia warstwa:

- Tyle neuronów ile klas

Funkcja straty:

- Cross-Entropy Loss

$$H(p, q) = \sum_i p_i(x) \cdot \log(q_i(x))$$

## W stronę klasyfikacji. Warstwa Softmax

- Podobieństwo do sigmoidy w regresji logistycznej

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_j^K e^{z_j}}$$

# Maximum Likelihood Principle

- Step by step:

$$p(data) = \prod_i p(c = grt_i \mid x_i)$$

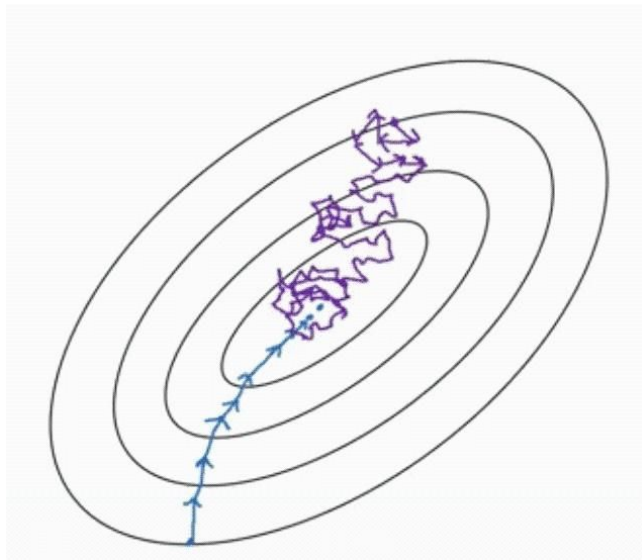
- Negative Log-Likelihood




$$-\ln p(data) = - \sum_i \ln p(c = grt_i \mid x_i)$$

$$p(c = grt_i \mid x_i) = \frac{e^{w \cdot x_i + b} grt_i}{\sum_j e^{w \cdot x_i + b}}$$

# Proces treningu

- Trenujemy w N epokach
- Batch GD vs. Stochastic GD



Deep learning model after....	
Epoch 1	
Epoch 10	
Epoch 99	

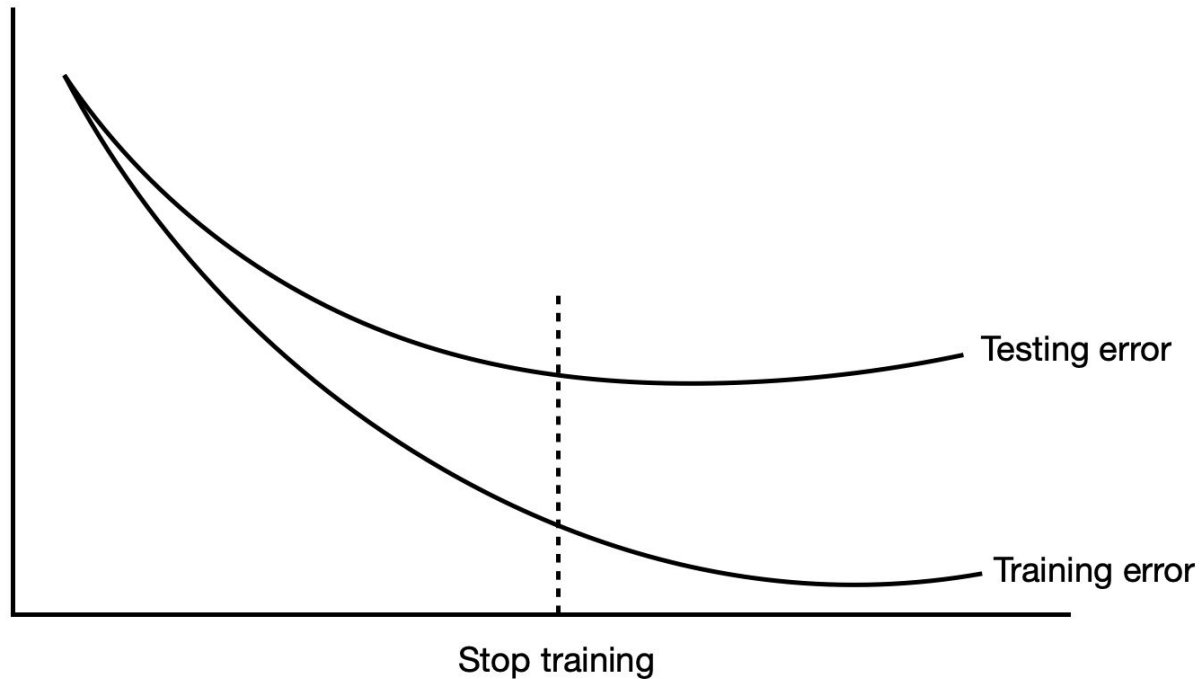
# Metody na przeuczenie

- Wczesne zatrzymanie trenowania
- Regularyzacje
- Dropout





# Wczesne zatrzymanie trenowania



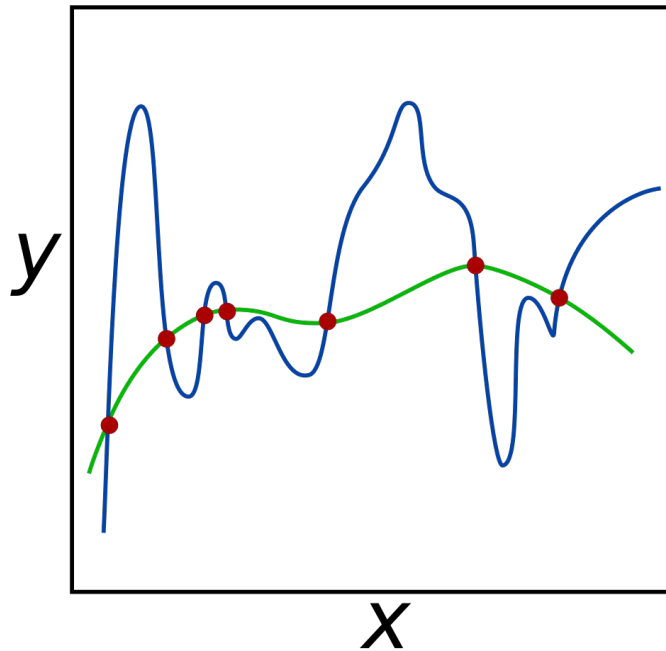
# Regularyzacja

- Regularyzacja L2

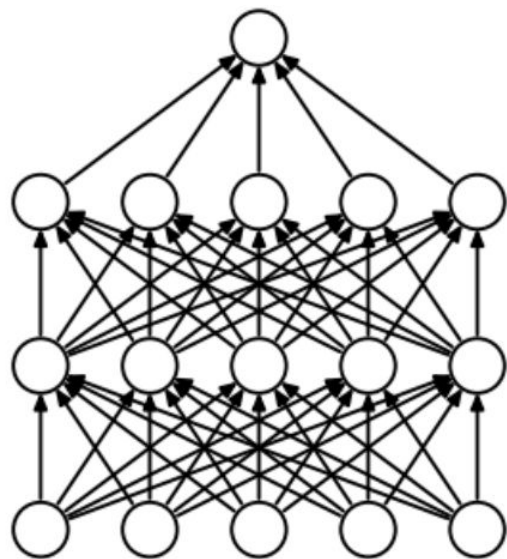
$$Loss := Loss + \lambda \cdot ||w||_2^2$$

- Regularyzacja L1

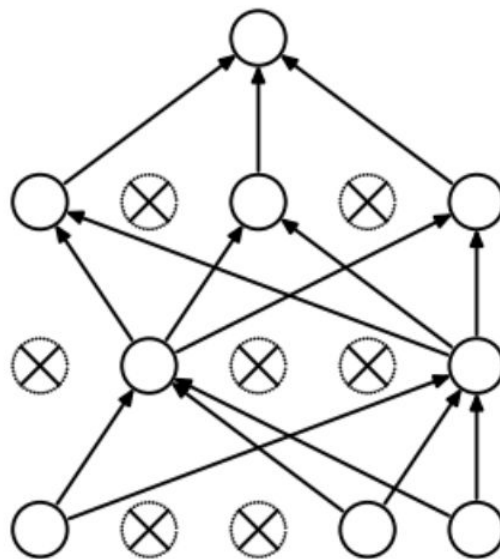
$$Loss := Loss + \lambda \cdot ||w||_1$$



# Dropout



(a) Standard Neural Net



(b) After applying dropout.

# Co by jeszcze poczytać?

- [Dropout](#)
- [Batch Normalization](#)
- [Inicjacja wag](#)

## Sources:

- <https://icon-icons.com/icon/pytorch-logo/169823>
- <https://www.gstatic.com/devrel-devsite/prod/v7824338a80ec44166704fb131e1860a66ed443b0ce02adfe8171907535d63bde/tensorflow/images/lockup.svg>
- <https://www.symmatrix.com/product/mxnet/>
- <https://www.oreilly.com/library/view/keras-2x-projects/9781789536645/8e433506-15b5-46d9-b7a4-240eeae6a0f8.xhtml>
- [https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function)
- <https://vitalflux.com/tensor-explained-with-python-numpy-examples/>
- [https://miro.medium.com/max/2400/1\\*iWQzxhVlvadk6VAJjsqXgg.png](https://miro.medium.com/max/2400/1*iWQzxhVlvadk6VAJjsqXgg.png)
- <https://i.ytimg.com/vi/5HNdxZFUB5w/maxresdefault.jpg>
-