```
activation: function
                          activation d: function
                          activation name : str
                          adam m: list
                          adam v : list
                          cuda available : bool
                          cuda max threads per block : int
                          cuda n streams : int
                          delta: numpy.ndarray
                         delta backward_input_: numpy.ndarray
                          delta backward output : numpy.ndarray
                          do backward function : function
                          do backward impl : str
                          do backward output function: function
                          do backward output impl:str
                          do forward function: function
                          do forward impl : str
                          gradient : list
                          initial: bool
                          input : numpy.ndarray
                          input shape : tuple
                          11 penalties : list
                          12 penalties : list
                          name : str
                          output : numpy.ndarray
                          output shape : tuple
                          prev correction: list
                          shortname default prefix : str
                          tunable : bool
                          weights: list
                            compile (...)
                           init (...)
                           setup forward backward impls (...)
                          backward(...)
                          do backward()
                          do backward output()
                          do forward()
                          forward(...)
                          relu()
                          relu d()
                          sigmoid()
                          sigmoid d()
                          softmax()
                          softmax d()
                          weights conv glorot normal(...)
                          weights conv glorot uniform(...)
                          weights conv he normal(...)
                          weights conv he uniform(...)
                          weights glorot normal(...)
                          weights glorot uniform(...)
                          weights he normal(...)
                          weights he uniform(...)
                                                                                                                                             SequentialClassifier(sklearn.base.BaseEstimator, sklearn.base.ClassifierMixin)
                                                                                                                                             adam beta 1 : float
                                                                                                                                             adam beta 2 : float
                                                                                                                                             adam epsilon: float
                                                                                                                                             class_labels_: numpy.ndarray
                                                                                                                                             decay rate : float
                                                                                                                                             fit_info_ : list
                                                                                                                                             fit_ongoing_: bool
                                                                                                                                             gradient_clip_: bool
                                                                                                           MaxPool2D
                                                                                                                                             layers_: list
                                                                                                                                             layers dict : dict
                                                                                         argmax : numpy.ndarray
                                                                                                                                             learning rate: float
                                                                                         height: int
                                                                                         n kernels: int
                                                                                                                                            learning rate now: float
                                                                                        n pools height : int
                                                                                                                                             loss : function
                                                                                                                                            loss d : function
                                                                                         n pools width: int
           Dense
                                                                                         pool size : int
                                                                                                                                            loss name : str
                                          Dropout
                                                                                         width : int
                                                                                                                                             momentum rate : float
                                                                      Flatten
       n neurons : int
                                  coeffs_: numpy.ndarray
                                                                                                                                             n batches: int
                                                                                           compile (...)
                                  model : NoneType
                                                                                                                                            n epochs: int
  compile (...)
                                                                                           init (...)
                                  rate: float
  init (...)
                                                                                                                                             use adam: bool
                                                                compile (...)
                                                                                         do backward()
do backward()
                                                                                                                                               init__(...)
                                    compile (...)
                                                                init (...)
                                                                                         do backward output()
do backward numpy()
                                                                                        do backward output numba cuda direct()
                                    init (...)
                                                              do backward()
                                                                                                                                             acc(...)
do backward output()
                                                                                        do backward output numba cuda direct job(...)
                                                                                                                                             add(...)
                                  do backward()
                                                              do backward output()
do_backward_output_numpy()
                                                                                        do backward output numba jit()
                                                              do forward()
                                  do backward output()
                                                                                                                                             backward(...)
do forward()
                                                                                                                                             categorical crossentropy(...)
                                                                                        do backward output numba jit job(...)
                                  do forward()
do forward numpy()
                                                                                        do backward output numpy()
                                                                                                                                             categorical crossentropy d(...)
                                                                                         do forward()
                                                                                                                                             fit(...)
                                                                                         do forward numba cuda direct()
                                                                                                                                             forward(...)
                                                                                         do forward numba cuda direct job(...)
                                                                                                                                             get weights()
                                                                                         do forward numba jit()
                                                                                                                                             loss(...)
                                                                                                                                             memorize fit info(...)
                                                                                         do forward numba jit job(...)
                                                                                        do forward numpy()
                                                                                                                                             predict(...)
                                                                                                                                             predict proba(...)
                                                                                                                                             print fit info(...)
                                                                                                                                             prune dev references()
                                                                                                                                             set weights(...)
                                                                                                                                             square loss(...)
                                                                                                                                             square loss d(...)
                                                                                                                                             summary()
                                                                                                                                             weights_li_norm()
                                                                                                                                             weights 12 norm()
```

Layer

Conv2D

dev weights 0 complex : DeviceNDArray

do backward numba cuda direct()

do backward numba cuda tiles()

do backward numba jit job(...)

do backward output numba jit()

do forward numba cuda direct()

do forward numba cuda tiles()

do forward numba cuda viafft()

do forward numba jit direct()

do forward numba jit gemm()

do forward numpy gemm()

numba cuda job fft(...)

numba\_cuda\_job\_r2c(...)
rules for promising impls()

do backward output numpy()

do forward()

do forward cv2()

do backward numba jit()

do backward numpy()

do backward output()

do backward numba cuda direct job(...)

do backward numba cuda tiles job(...)

do\_backward\_output\_numba\_cuda\_direct()

do backward output numba cuda tiles()

do backward output numba cuda viafft()

do backward output numba jit job(...)

do forward numba cuda direct job(...)

do forward numba cuda tiles job(...)

do forward numba jit direct job(...)

do forward numba jit gemm job(...)

numba cuda job c2rsame in do backward output(...)

numba\_cuda\_job\_muladdffts\_in\_do\_backward\_output(...)

numba cuda job c2rsame in do forward(...)

numba cuda job muladdffts in do forward(...)

do backward output numba cuda direct job(...)

do backward output numba cuda tiles job(...)

height\_: int kernel size : int

width: int

n channels : int

compile (...)

n kernels: int

\_\_init\_\_\_(...)

do backward()