```python
from tensorflow.keras.models import Model
from tensorflow.keras import regularizers
from tensorflow.keras.layers import (
    Dense,
    Conv2D,
    Dropout,
    Flatten,
    BatchNormalization,
    Activation,
    Add,
    Input,
    ZeroPadding2D,
    AveragePooling2D,
)
from tensorflow.python.keras.layers.core import SpatialDropout2D
from tensorflow.keras.layers.experimental.preprocessing import RandomCrop, Rando
mFlip
from tensorflow.python.keras.layers.preprocessing.image_preprocessing import HOR
IZONTAL
import numpy as np

BOTTLENECK = False  # enable this for CIFAR-100, sorry for the bad code design :
/

# https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scra
tch-in-tensorflow/
# https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Resi
dual_Learning_CVPR_2016_paper.pdf
def basic_blk(input, k, f, s):
    out = BatchNormalization(axis=3, momentum=0.9)(input)
    out = Activation("elu")(out)
    out = Conv2D(
        filters=f,
        kernel_size=k[0],
        kernel_initializer="he_normal",
        kernel_regularizer=regularizers.l2(l2=0.0001),
        padding="same",
        strides=s,
    )(out)

    out = BatchNormalization(axis=3, momentum=0.9)(out)
    out = Activation("elu")(out)
    out = Conv2D(
        filters=f,
        kernel_size=k[1],
        kernel_initializer="he_normal",
        kernel_regularizer=regularizers.l2(l2=0.0001),
        padding="same",
    )(out)

    if BOTTLENECK:
        out = BatchNormalization(axis=3, momentum=0.9)(out)
        out = Activation("elu")(out)
        out = Conv2D(
            filters=(4 * f),
            kernel_size=k[2],
            kernel_initializer="he_normal",
            kernel_regularizer=regularizers.l2(l2=0.00001),
            padding="same",
        )(out)
    return out
```

```python

def ident_blk(input, filter_depth):
    ff_input = input
    out = Dropout(0.5)(input)
    out = basic_blk(out, (3, 3), filter_depth, (1, 1))
    out = Add()([out, ff_input])
    out = SpatialDropout2D(0.5)(out)
    return out


def conv_blk(input, filter_depth, stride):
    ff_input = input
    out = basic_blk(input, (3, 3), filter_depth, stride)
    ff_input = BatchNormalization(axis=3, momentum=0.9)(ff_input)
    ff_input = Activation("elu")(ff_input)
    ff_input = Conv2D(
        filter_depth,
        kernel_size=(1, 1),
        kernel_initializer="he_normal",
        kernel_regularizer=regularizers.l2(l2=0.00001),
        strides=stride,
        padding="same",
    )(ff_input)
    out = Add()([out, ff_input])
    return out


def res_blk(x, filter_depth, num_layers, init_stride):
    x = conv_blk(x, filter_depth, init_stride)
    for i in range(num_layers - 1):
        x = ident_blk(x, filter_depth)
    return x


def ResNet_N(in_shape, layers, classes):
    filter_depth = 64
    input = Input(in_shape)

    # Preprocessing method: RANDOM CROP
    x = ZeroPadding2D(padding=(4, 4))(input)
    x = RandomCrop(32, 32)(x)
    x = RandomFlip(mode=HORIZONTAL)(x)

    # model
    x = Conv2D(
        filter_depth,
        kernel_size=3,
        kernel_initializer="he_normal",
        kernel_regularizer=regularizers.l2(l2=0.00001),
        padding="same",
        strides=2,
    )(x)

    x = BatchNormalization(axis=3, momentum=0.9)(x)
    x = Activation("elu")(x)

    x = res_blk(x, filter_depth, layers[0], init_stride=1)
    for i in range(len(layers[1:])):
        x = res_blk(x, (2 ** (i + 1)) * filter_depth, layers[i + 1], init_stride
=2)

    x = AveragePooling2D(padding="same")(x)
    x = Flatten()(x)
    x = Dropout(0.5)(x)
    x = Dense(classes, activation="softmax", kernel_initializer="he_normal")(x)
    model = Model(
        inputs=input, outputs=x, name=("ResNet-" + str(2 * np.sum(layers) + 2))
    )

    return model
```