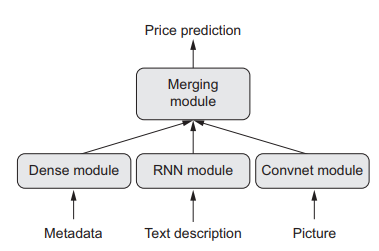
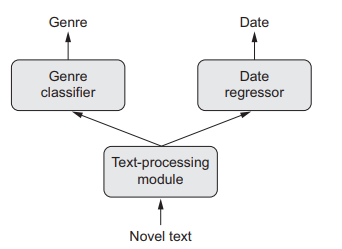
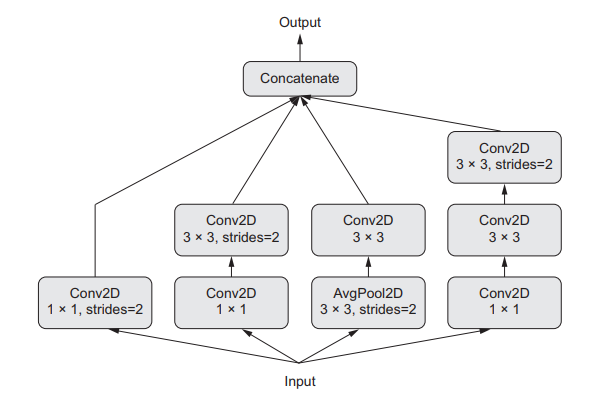
|  |  |
| --- | --- |
| **metadata** | **Dense MLP** |
| **Dest text** | **RNN** |
| **Img** | **CNN** |

****

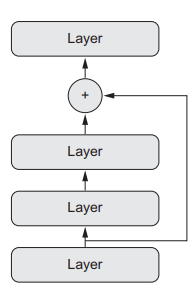
Merging input model

****

Muliti task model

****

Inception model

****

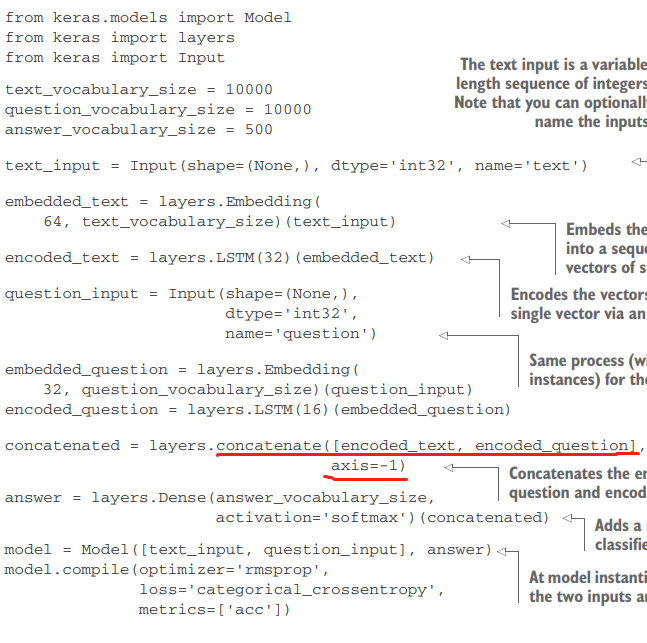
Residual model

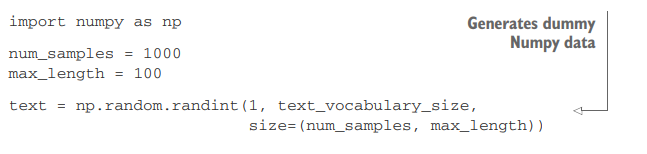
**Multi inout**

**Multi output**

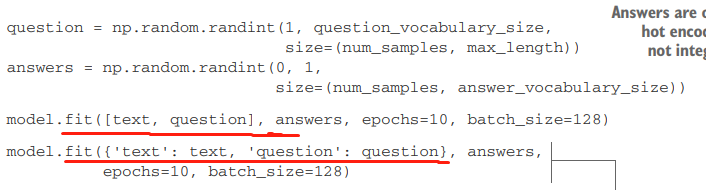
**Graphlike**

**Multi Input Example**

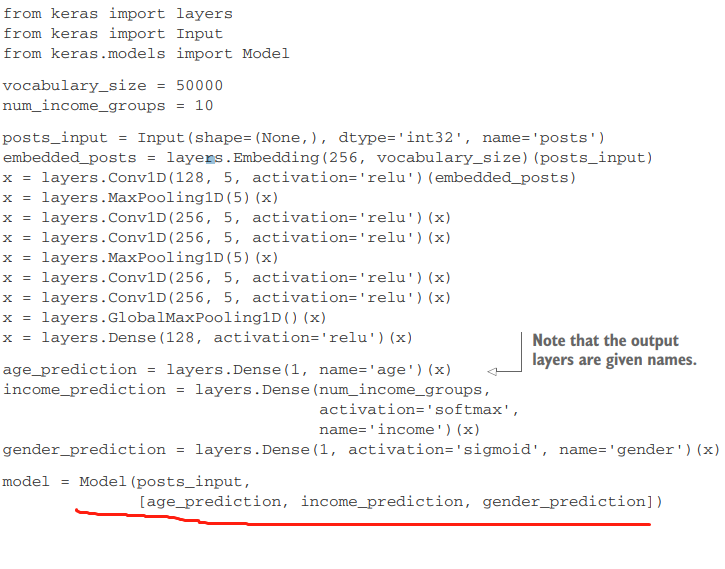
****

****

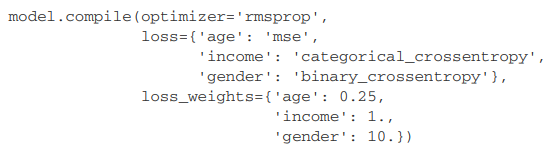
**Fit**

****

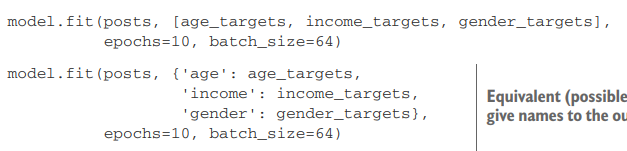
**Multi output example**

****

**Multi loss for Multi output**

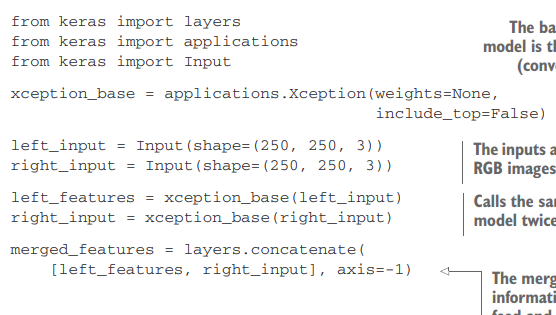
****

**Fit**

****

**Model layer tensor great example**

**Reuse the xception layer**

****

批正则化

为了减少输入的不同样本之间的数据差异对于模型学习的影响，目前普遍采用的策略之一便是归一化，对不同大小的数据进行归一化操作之后，往往能够使得模型在新数据上的泛化能力更强。最常见的归一化手段通常是先用数据均值将数据的中心置零，然后再通过除以数据的标准差，将数据的大小范围统一。深度学习早期研究中，通常是在将数据送入模型训练之前对输入数据进行归一化，但实际上数据归一化操作应当在网络的每一层输入前都应该执行一次。因为即使保证了输入数据是0均值和单位标准差的，也无法保证之后的每一层输入都是0均值和单位标准差。为此，Batch Normalization由Ioffe和Szegedy在2015年提出，它的引入，使得网络能够在训练过程中自适应地进行中间层数据的归一化。它的原理是通过

内部计算每一个batch的数据维持均值和方差，给深度学习带来的好处就是更加有利于反向传播梯度优化，从而能够训练更深的网络。对于一些非常深的常见深度网络模型，往往都含有batch normazation层，比如Resnet50, Inceptionv3和xception。

深度可分卷积

与通常的卷积操作不同，depthwise separable convolution并没有对所有通道统一进行卷积操作，而是对每个通道分别进行卷积操作，得到每个通道的特征图，然后再利用单像素卷积操作，对得到的联合通道特征图进行卷积操作，最终获得该层新的特征输出。它的原理是假设对于每个通道而言，具有空间局部相关性，而对于不同通道来说，通道之间不具有相关性。通过逐通道进行卷积而不是所有通道一起卷积，可以使得训练参数和计算量大大减小，从而获得更小，但更快的模型，并且由于其卷积方式更具有表达性，往往能够用更少的数据取得更好的效果。这些优点对于在小样本集上训练得到小模型的场景非常有帮助。采用该卷积方式的深度卷积神经网络模型有Xception等。

超参优化

当构建深度学习网络模型时，往往面临许多参数选择问题，包括学习速率选择、激活函数选择、层数、每层隐藏单元数、Dropout参数等等，这些参数选择的过程称为超参优化。关于超参优化并没有比较明确的原则指导，往往根据具体样本和经验而定，甚至带有一定随机性。为了进行参数优良性对比，通常是实验观察不同参数的模型在validation验证集上的分类表现。

集成学习

集成学习的主要思想是基于假设：不同的模型能够学习到不同样本特征，能够从不同角度来描述样本，因此，综合多个不同的模型，可以获得比单个模型更加全面，从而更好的结果。综合策略可以分为以下几种途径：1）第一种，将所有模型的输出进行简单加权，直接采用求平均值本质上会忽略不同分类模型的优劣，因此可采用的加权策略如Nelder-Mead策略，即：



，其中由不同模型的结果根据以上对应权重加权而成。集成学习的关键是确保模型之间具有很高的多样性。最近，已有一些研究尝试将深度模型和浅层模型融合，并取得不错的效果。

过拟合

过拟合是深度学习面临的主要问题之一，对于分类问题而言，是指在一定迭代次数之后，随着继续训练迭代，模型在训练集上的分类准确率不断提高，但是在验证集上的分类准确率却出现下降，原因是模型过于拟合训练集的分布，从而学习到了许多训练集特有而验证集没有的噪声特征，导致模型泛化能力变差。减轻过拟合一般有以下几种策略：1）使用更少的网络参数，包括减少网络层数、减少层内隐藏单元数等；2）添加正则项，通过为权重添加惩罚函数，使得网络倾向于选择较小的权重，往往能够得到泛化效果更好的网络模型，惩罚函数通常可以分为L1 regularization和L2 regularization两种；3）添加Dropout层，按照参数概率随机丢弃一部分中间输出；4）引入BatchNormalization层，使得每一层输出都进行归一化处理。