基于改进核 CNN 的离线孤立手写汉字识别

杜金鸿

2017.05.01

摘要

针对孤立手写汉字的离线识别问题，借鉴 GoogleNet 的思想，利用深度学习开源框架 Tensorflow 搭建了一种广度与深度相平衡的卷积神经网络模型，使用改进的横特征、竖特征矩形卷积核，并给出其网络拓扑结构，通过随机旋转、随机裁剪、随机添加高斯噪声对 HCL2000 样本集进行扩展，在增加样本量的同时提高了模型的泛化能力，对产生仿射形变、弹性形变等的手写汉字样本有较好识别效果。在测试集数据的测试中，可达到 97% 以上的正确率，从而验证了文中识别方法的有效性。最后利用 Django 搭建了孤立手写汉字识别的 Web 应用，使模型得以投入实际应用。

关键词 手写汉字识别 卷积神经网络 矩形卷积核 仿射形变 弹性形变 高斯噪声 Web 应用

1 综述

1.1 手写汉字识别

手写体字符识别是模式识别领域的一个难点与热点，传统识别算法结合结构特征和统计特征进行识别，特征的人工选择与提取显得尤其重要。离线的手写汉字不包含汉字笔画书写顺序的信息，对其识别更加困难。尽管目前离线的孤立手写汉字识别在某些数据集上已能达到 95% 以上的识别率，但在实际应用中，识别正确率往往明显下降。影响识别的因素主要有：噪声、拓扑结构变化、形近字等。

本文利用 HCL2000[1] 数据集进行训练与预测，包含了 3755 个国家一级常用字，每个汉字有 1000 个样本，每个样本为 64×64（pixels）的灰度图像，部分样本图像如下：

1.2 深度学习

深度学习模仿人类大脑工作过程，对输入数据（图像、音频等）进行识别学习。人类识别一张图片上的物体为狗的过程是：识别特定部位，例如鼻子；再进一步地，为了确定某一区域是不是鼻子，我们可以将其再分解为多个小区域，观察它的外轮廓形状、鼻孔形状等。也就是说，我们通过较简单的特征，拼合较高阶特征，并利用高阶特征来对整体作出判断。这正是卷积神经网络（Convolutional Neural Network，CNN）所会学习的，并且 CNN 是通过正向、反向传播修正误差的方式来学习的。此外，CNN 还具有局部感受野、权值共享和次抽样等结构特性。

目前，已有不少学者对基于深度学习的手写汉字离线识别进行研究并取得不错的效果 [2]，但是对相似字、形变字的识别效果仍有提升空间。通常对 CNN 的改进主要着眼于网络结构、训练算法两方面。本文利用长宽不同的卷积核强化横、竖特征的提取，并对训练集进行形变、加噪处理，提升 CNN 模型的泛化能力，最后构建了手写汉字识别的 Web 应用。

2 神经网络结构

2.1 Inception

深度神经网络分级抽象学习的能力依赖于网络深度；GoogleNet 利用 Inception 结构，扩展了网络的广度 [3]。本文平衡网络的深度与广度，模仿 GoogleNet 的 Inception 结构，在每一个 Inception 内，CNN 利用不同的卷积核对张量（Tensor）进行特征提取，在 Inception 之后将所提取的特征进行拼接，传入后续网络。

2×2 卷积核：

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \xrightarrow{\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}}$$

$$\begin{pmatrix} x_{11}a_{11}+x_{12}a_{12}+x_{21}a_{21}+x_{22}a_{22} & x_{11}a_{12}+x_{12}a_{13}+x_{21}a_{22}+x_{22}a_{23} & x_{11}a_{13}+x_{12}a_{14}+x_{21}a_{23}+x_{22}a_{24} \\ x_{11}a_{21}+x_{12}a_{22}+x_{21}a_{31}+x_{22}a_{32} & x_{11}a_{22}+x_{12}a_{23}+x_{21}a_{32}+x_{22}a_{33} & x_{11}a_{23}+x_{12}a_{24}+x_{21}a_{33}+x_{22}a_{34} \\ x_{11}a_{31}+x_{12}a_{32}+x_{21}a_{41}+x_{22}a_{42} & x_{11}a_{32}+x_{12}a_{33}+x_{21}a_{42}+x_{22}a_{43} & x_{11}a_{33}+x_{12}a_{34}+x_{21}a_{43}+x_{22}a_{44} \end{pmatrix}$$

$2\times 1 \ + \ 1\times 2$ 卷积核：

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \xrightarrow{\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}} \begin{pmatrix} y_1a_{11}+y_2a_{21} & y_1a_{12}+y_2a_{22} & y_1a_{13}+y_2a_{23} & y_1a_{14}+y_2a_{24} \\ y_1a_{21}+y_2a_{31} & y_1a_{22}+y_2a_{32} & y_1a_{23}+y_2a_{23} & y_1a_{24}+y_2a_{34} \\ y_1a_{31}+y_2a_{41} & y_1a_{32}+y_2a_{42} & y_1a_{33}+y_2a_{43} & y_1a_{34}+y_2a_{44} \end{pmatrix} \xrightarrow{\begin{pmatrix} z_1 & z_2 \end{pmatrix}}$$

$$\begin{pmatrix} z_1y_1a_{11}+z_1y_2a_{21}+z_2y_1a_{12}+z_2y_2a_{22} & z_1y_1a_{12}+z_1y_2a_{22}+z_2y_1a_{13}+z_2y_2a_{23} & z_1y_1a_{13}+z_1y_2a_{23}+z_2y_1a_{14}+z_2y_2a_{24} \\ z_1y_1a_{21}+z_1y_2a_{31}+z_2y_1a_{22}+z_2y_2a_{32} & z_1y_1a_{22}+z_1y_2a_{32}+z_2y_1a_{23}+z_2y_2a_{33} & z_1y_1a_{23}+z_1y_2a_{33}+z_2y_1a_{24}+z_2y_2a_{34} \\ z_1y_1a_{31}+z_1y_2a_{41}+z_2y_1a_{32}+z_2y_2a_{42} & z_1y_1a_{32}+z_1y_2a_{42}+z_2y_1a_{33}+z_2y_2a_{43} & z_1y_1a_{33}+z_1y_2a_{43}+z_2y_1a_{34}+z_2y_2a_{44} \end{pmatrix}$$

在实际应用中，卷积核的形状通常设定为正方形：$1\times 1, 3\times 3, \cdots$ 长宽不等的卷积核一般仅用于拉伸形变图像的处理。如上图所示，直接采用 2×2 卷积、先进行 2×1 再进行 1×2 卷积（卷积步长均为 1），对优化参数的数目没有影响。但是如果在后者两次卷积过程之间再加上其他操作（例如最大池化），或者改变卷积的步长（例如控制纵向、横向步长不同），两者的效果就不同了。

若直接进行 2×2 卷积，则该过程更倾向于关注卷积核所在区域的整体情况；若在 2×1 卷积后采用最大池化，则该过程更倾向于关注图像数据的行特征。例如当输入张量为

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

若采用 2×2 卷积，步长为 1，则输出：

$$\begin{pmatrix} x_{12}+x_{22} & x_{11}+x_{21} & 0 \\ x_{22}+x_{32} & x_{21}+x_{31} & 0 \\ x_{32}+x_{42} & x_{31}+x_{41} & 0 \end{pmatrix}$$

若先采用 2×1 卷积，步长为 2×1，则输出：

$$\begin{pmatrix} 0 & y_1+y_2 & 0 & 0 \\ 0 & y_1+y_2 & 0 & 0 \end{pmatrix}$$

再采用最大池化，滤波器大小为 2×2，步长为 1（不足则填充像素）：

$$\begin{pmatrix} \max\{0, y_1+y_2\} & \max\{0, y_1+y_2\} & 0 \end{pmatrix}$$

最后进行 1×2 卷积，步长为 1×2，最终得到：

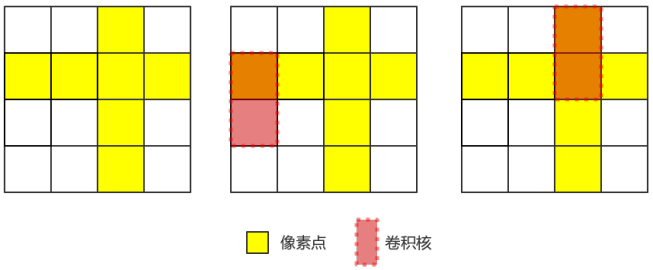
$$\begin{pmatrix} z_1 \max\{0, y_1+y_2\} & 0 \end{pmatrix}$$

可以看到，从直观上理解，后者相当与对图片先做纵向压缩处理，再在压缩后的图片各行间寻找模式特征。

CNN 的深度学习在于特征的层次表示，即 CNN 会从浅层的特征开始学习，例如：点、线、面等等，再将其拼接组合形成较高阶的特征。由于训练样本汉字图像基本接近二值化，即汉字部分像素值接近 255，而背景部分像素值几乎为 0，只有汉字与背景的交界处像素值呈过渡状，因此手写汉字在以正方形为卷积核的 CNN 中

被抽象出的浅层特征将主要是汉字的边缘特征。这是因为在背景或者汉字内部卷积核的作用几乎是一样的（汉字笔画像素宽度基本大于等于卷积核边长）。因此采用传统的卷积核对差别极小的形近字，例如“子”和“予”，仅仅提取边缘特征，可能难以准确识别。

采用长宽不等的卷积核，如下图所示，采用了 2×1 的卷积核，对于竖笔画内部，卷积核的作用几乎一致，但在其它笔画的像素区域该卷积核会主动学习其边缘特征，也就是说，该卷积核加强对非竖笔画的边缘特征提取。同样的，1×2 的卷积核可以加强对非横笔画的边缘特征提取。方便起见，本文将上述两类卷积核分别称为横、竖特征提取卷积核。



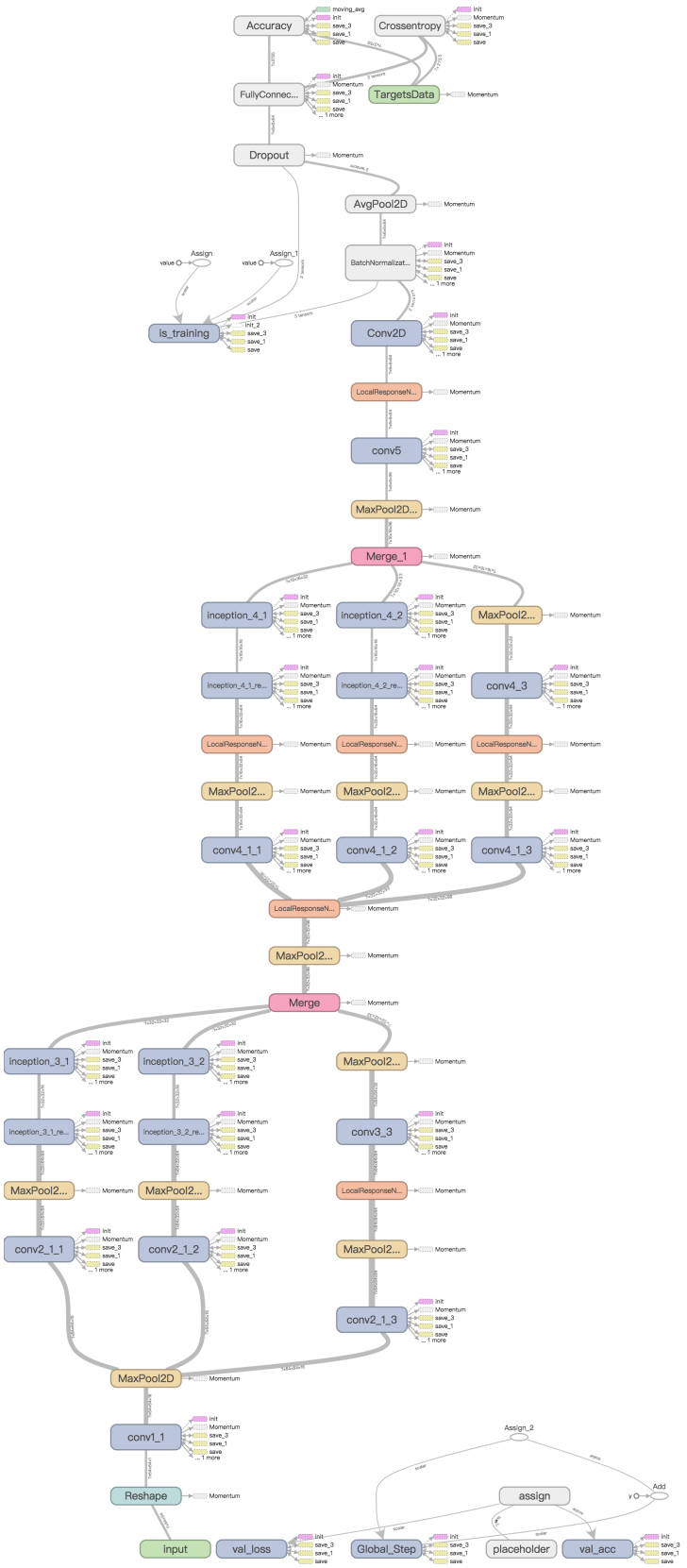
与普通正方形卷积核的 CNN 相比，利用横特征、竖特征卷积核进行卷积的方式对于弹性形变文字的识别也是有益的。例如，经过纵向压缩处理的图像与原图像利用横特征卷积核提取的特征图相似度会高于利用普通正方形卷积核提取的特征图。

考虑到汉字中的笔画特点，横、竖居多，对于撇捺等笔画，其实也可以理解为大小、长短不同的横、竖笔画拼接得到，因此在孤立手写汉字灰度图像识别这个特定情形下，我们完全可以充分利用这些横、竖特征，以改善 CNN 的识别效果。

因此，本文在每个 Inception 中分别配置了横特征、竖特征提取 CNN 以及一个普通 CNN（卷积核为正方形）。在每个横特征、竖特征提取 CNN 之后还需加上相应的竖特征、横特征提取 CNN，这是为了规范输出张量的形状。

完整的深度神经网络拓扑结构如右图所示。

需要指出的是，在输入、输出张量大小相同的情况下，采用上述卷积核以及只采用常规卷积核的参数数量也可能不同，即使网络仅卷积核大小和步长有差异、卷积层数和池化层等均完全一致。



2.2 Normalization

归一化的目的是为了解决某一层导致的输出空间与目标空间分布的不一致性，防止梯度弥散。

Local Response Normalization 通过公式 (1) 进行更新，

$$x = \frac{x}{\left(bias_{LRN} + \frac{\alpha_{LRN}}{d_{LRN}} \sum_i x_i^2\right)^{\beta_{LRN}}} \tag{1}$$

其中 $bias_{LRN}$ 为偏置项，d_{LRN} 为归一化窗口边长的一半，α_{LRN} 为缩放因子，β_{LRN} 为指数项。其中求和项是对归一化窗口内的数据求和，不足则填充像素。

Batch Normalization[4] 方法是比 Local Response Normalization 是一种更为有效的方法。它的优点有：提高学习效率、改善梯度计算、减少对初始化权重的依赖、替代 dropout 的使用。但同时也有文章指出使用 Batch Normalization 会带来 30% 额外的计算开销。其更新算法如下 [4]：

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

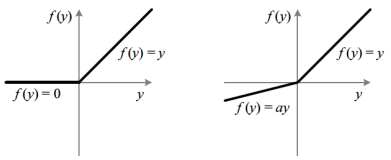
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

其中 β 偏置项,γ 为缩放因子,ε 为偏置项（防止出现分母为 0 的情况）,*decay* 为权值衰减率,*stddev* 为截断步长。

本文综合使用了上述两种归一化层。

2.3 Activation Function



Rectified Linear Unit（Relu）是常用的激活函数，其函数图像如上图。Parametric Rectified Linear Unit（PRelu）是 Relu 的推广 [5]，它在自变量取值为负的部分用线性函数代替常值函数，避免了当 $y \leqslant 0$ 时，误差后向传播梯度为 0，即死亡神经元的情形。本文采用 PRelu 作为激活函数。

在最后的全连接层处，本文采用传统的 *softmax* 多分类器。*softmax* 函数和 *sigmoid* 函数一样，每个单元的输出会被压缩到 0 和 1 之间，并且输出之和为 1。不同的是，*softmax* 可以处理多类别问题，其输出可以看作一个表示正确分类的概率分布，

$$\sigma(x) = \frac{1}{\sum_{i=1}^{n} e^{x_i}}(e^{x_1}, \cdots, e^{x_n})^T \tag{2}$$

其中 $x = (x_1, \cdots, x_n)^T$。

2.4 Optimization

本文所使用的损失函数为分类交叉熵，

$$C(\hat{y}, y) = -\sum_{i=1}^{n} y_i \ln \hat{y}_i \tag{3}$$

其中 $y = (y_1, \cdots, y_n)^T$ 为样本标签,$\hat{y} = (\hat{y_1}, \cdots, \hat{y_n})^T$ 为网络预测标签。y 是利用 $One-hot$ 编码将样本标签（$0, 1, \cdots,$ 3754）转换为长度为 3755 的 $0-1$ 向量。

本文对损失函数的优化利用 *Momentum* 梯度下降法 [6]，

$$w = w - \eta \nabla Q_i(w) + \alpha \Delta w \tag{4}$$

其中 η 为学习率，α 为人为设定的超参数，w 为梯度值，$\nabla Q_i(w)$ 为当前梯度方向。梯度更新公式依据物理学势能与动能之间能量转换关系，当动量（momentum）越大，转换为势能的能量越大，更有机会摆脱局部极值，进入全局凸域，通过这种方式可以检索修改方向加快收敛速度。

2.5 Dropout

Dropout 是防止过拟合的有效方法 [7]，在每一次网络某一层正向传播过程中，通过随机丢弃部分神经元的前后连接实现。设神经元保留个数的概率为 *keep_prob*，则被保留神经元的输出需乘上 $\frac{1}{keep\_prob}$ 以保证输出值之和的期望不变。

在实际应用中，训练时会将 β 设定在 0.5 左右，在测试时则设定为 1.0 以最大化模型的能力。

3 训练过程

训练前先对输入数据进行零均值化、归一化处理，目的是防止奇异样本数据对训练收敛时间的影响。

深度神经网络的训练过程基本在配置 NAVIDA TITAN X(Pascal) 的 Ubuntu16.04 平台上，利用 Google

深度学习开源框架 Tensorflow 进行。由于样本量巨大，训练时间久，故本文进行训练时未设定验证集。设定训练集、测试集比率为 9 : 1，Batch Size=32，学习率 $\eta = 0.001$, 动量权重 $\alpha = 0.9$, Dropout 保留概率 $keep\_prob = 0.6$, $d_{LRN} = 5$, $bias_{LRN} = 1.0$, $\alpha_{LRN} = 0.0001$, $\beta_{LRN} = 0.5$, $\beta = 0.0$, $\gamma = 1.0$, $\varepsilon = 1e-05$, $decay = 0.9$, $stddev = 0.002$。

在完全训练模型之前，本文先对基于改进核的 CNN 模型与常规核 CNN 模型进行了试探性训练，以比较两者性能的差异。为比较二者差异，将本文模型中每个 Inception 所使用的横特征提取核——大小 4×2、步长 $(2,1)$ 以及大小 1×2、步长 $(1,2)$、竖特征提取核———大小 2×4、步长 $(1,2)$ 以及大小 2×1，步长 $(2,1)$，替换为常规卷积核：大小 (2×2)、步长 $(1,1)$ 以及大小 (2×2)、步长 $(2,2)$，在每个 Inception 中，输出张量的形状保持不变。在不添加 Dropout 的情况下，对原训练集迭代一次，结果如下：

卷积核	Loss(训练集)	Accuracy(训练集)
普通核	0.32800	0.9327
改进核	0.25834	0.9453

卷积核	Accuracy(测试集)	训练时间
普通核	0.934511	10710.159s
改进核	0.943662	8626.966s

可以看到，改进核的 CNN 训练时间明显较短，并且在训练集、测试集上的表现均优于普通核。

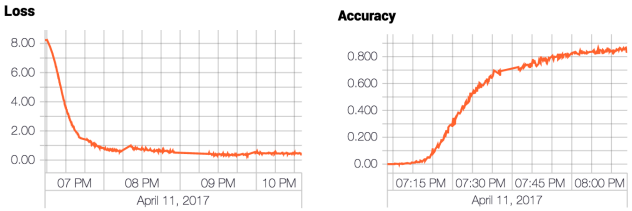
为对本文模型进行较充分训练，同时保证模型的泛化能力，添加 Dropout 层并且对输入图像进行各种形变、噪声处理。由于时间限制，且数据量巨大，仅对训练集进行以下 4 个 Epoch 迭代训练：

Epoch	操作
1	无
2	随机旋转（$\leqslant 20°$）
3	随机裁剪
4	高斯噪声（$\sigma = 0.3$）

上述各种处理不仅具有扩充样本的功能，还有增加噪声、防止网络过拟合的效果。随机旋转处理相当于对文字进行仿射形变处理，随机裁剪处理相当于对文字进行平移变换处理；高斯噪声处理训练增强网络的抗干扰能力。

4 结果分析

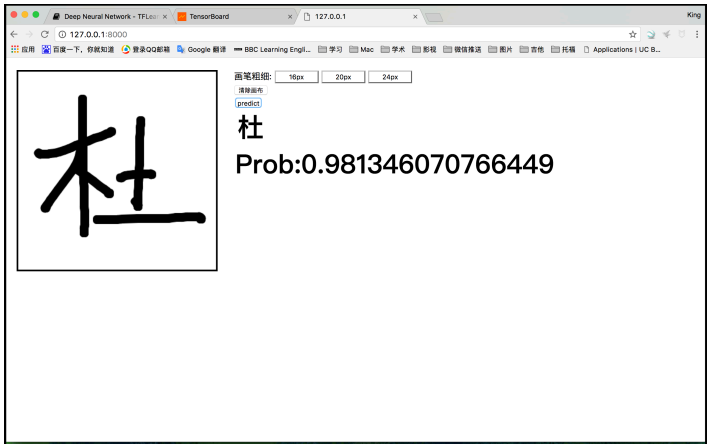
利用 Tensorflow 自带的可视化工具 TensorBoard 进行数据可视化，前期的训练集损失（loss）以及准确度（accuracy）如下图所示：



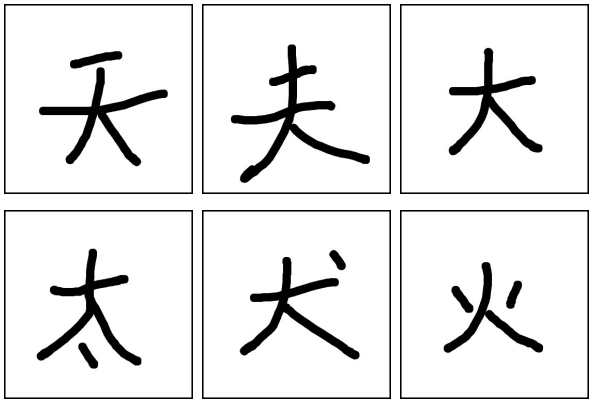
上图展示的仅为训练初期的变化曲线，可以看到初期收敛速度较快。最终经过四个 Epoch 迭代，模型在测试集上准确度可以达到 97.4405%。由于设备与时间限制，显然模型的训练仍不充分，还有提升空间。

已训练完成的深度学习模型可以完整保存下来，便于后续继续训练优化模型，也可将各权值利用迁移学习改造优化模型。

在深度神经网络模型训练完成后，本文利用 Django 搭建简易 Web 应用，实现在 Web 上利用鼠标输入孤立手写汉字、进行离线识别并反馈识别结果。界面如下：



对以下一组形近字进行识别（识别前需借助程序对图像进行反相以符合模型输入要求）：



结果如下：

识别汉字	预测汉字	概率
天	天	0.8455
夫	夫	0.9831
大	大	0.9206
太	太	0.9745
犬	犬	0.9743
火	火	0.9555

从上述结果可见，模型对形近字的识别较为准确（实际预测结果与书写的笔画粗细、整洁程度、书写习惯等多种因素有关）。

5 结论

本文将 CNN 应用于孤立手写汉字的识别，并采用仿射形变、弹性形变等方法扩展了样本集，从而提高了深度神经网络的泛化能力。实验结果表明：基于改进核的 CNN 性能优于基于普通卷积核的 CNN，深度神经网络模型对原测试样本、仿射形变测试样本、弹性形变测试样本和添加高斯噪声样本的预测准确度高。最后本文将本模型投入实际应用，搭建了孤立手写汉字识别的 Web 应用，识别效果优秀。

文中的深度神经网络模型训练仍不充分，仍有扩展、提升空间；同时训练过程中，网络权值均随机初始化、网络参数均选择常用值，而权值、参数对网络的收敛速度以及识别结果均有重要影响；另外，模型的训练依赖于训练样本，若能够增加新样本，在新数据集上继续训练，模型的泛化能力将大大提高。故可以考虑从这三方面进行改进提升网络能力。

6 参考文献

[1] http://www.pris.net.cn/download/hcl2000.

[2] 高学, 王有旺, 基于 CNN 和随机弹性形变的相似手写汉字识别, 华南理工大学学报 (自然科学版), 2014, (01):72-76+83.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going Deeper with Convolutions, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, 1-9.

[3] Ioffe, Sergey and Szegedy, Christian, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv*: 1502.03167, 2015.

[4] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *Proceedings of the IEEE international conference on computer*, 2015, 1026-1034.

[5] Sutskever, Ilya and Martens, James and Dahl, George and Hinton, Geoffrey, On the importance of initialization and momentum in deep learning, *In Proceedings of the* 30*th international conference on machine learning* pp.1139–1147, 2013.

[6] Srivastava, Nitish and Hinton, Geoffrey E and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, Dropout: a simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, 1929-1958, 2014.