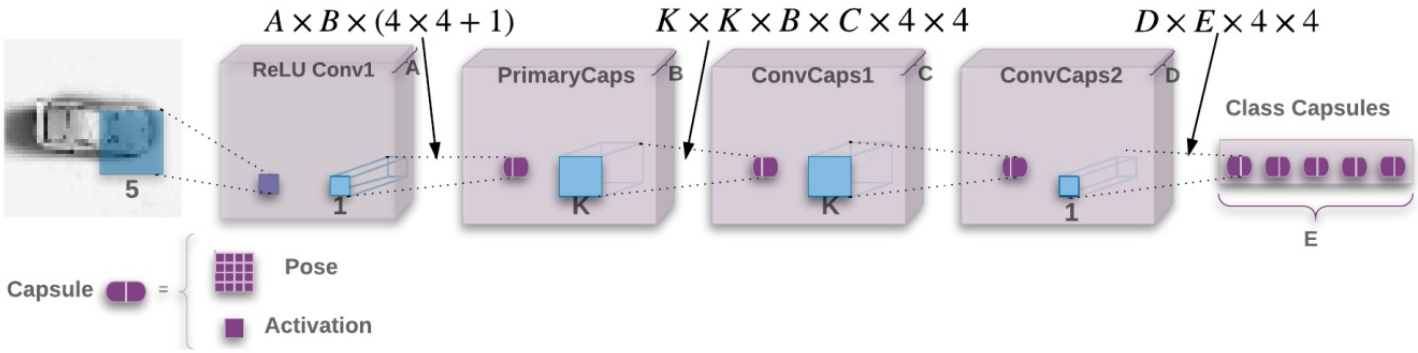
知乎 首发于 SCUT神经网络-Magellan小分队



CapsulesNet 的解析及整理

Nango ⋯

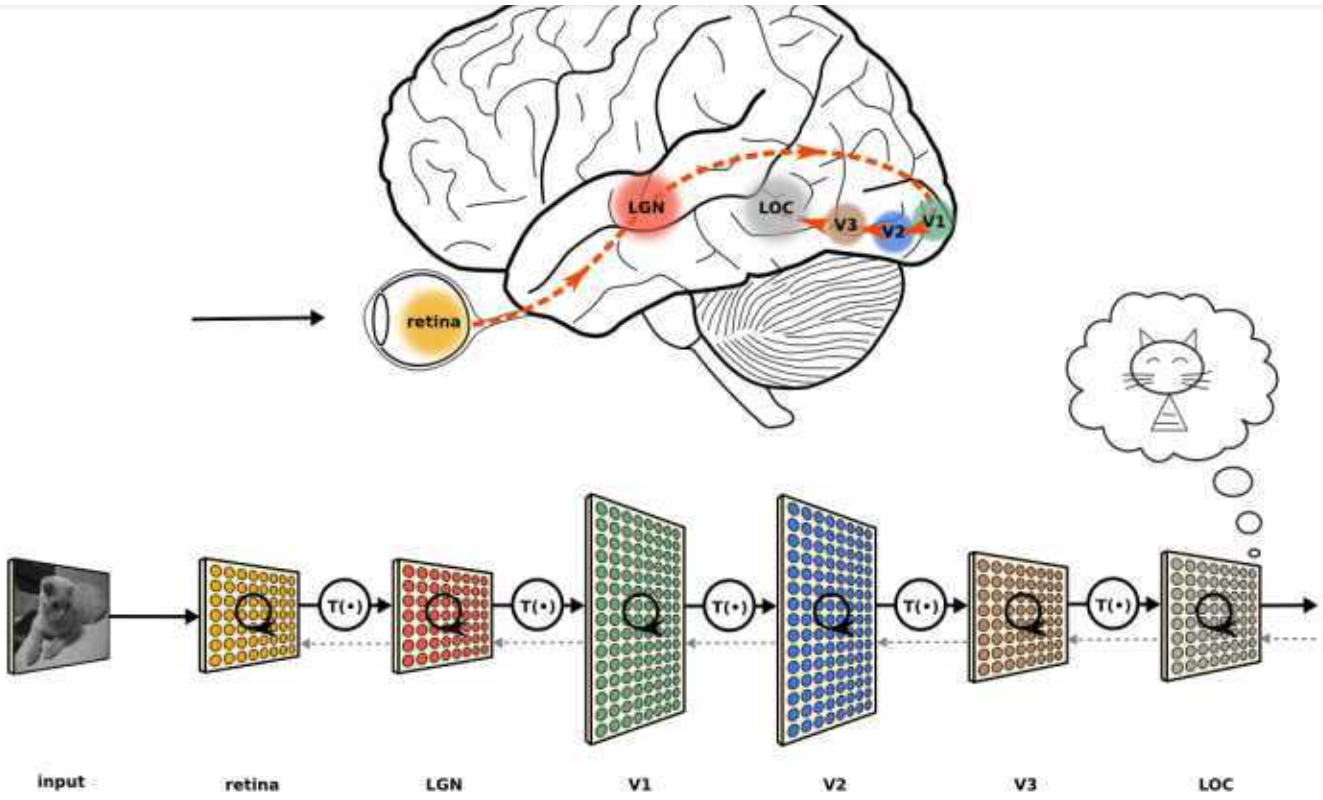
华南理工大学 计算机硕士，喜欢划水

参考：

- Dynamic Routing Between Capsules
- Matrix Capsules with EM routing
- 浅析 Hinton 最近提出的 Capsule 计划
- Capsule Networks Explained
- 先读懂CapsNet架构然后用TensorFlow实现：全面解析Hinton的提出的Capsule

Hinton 对CNN的思考：

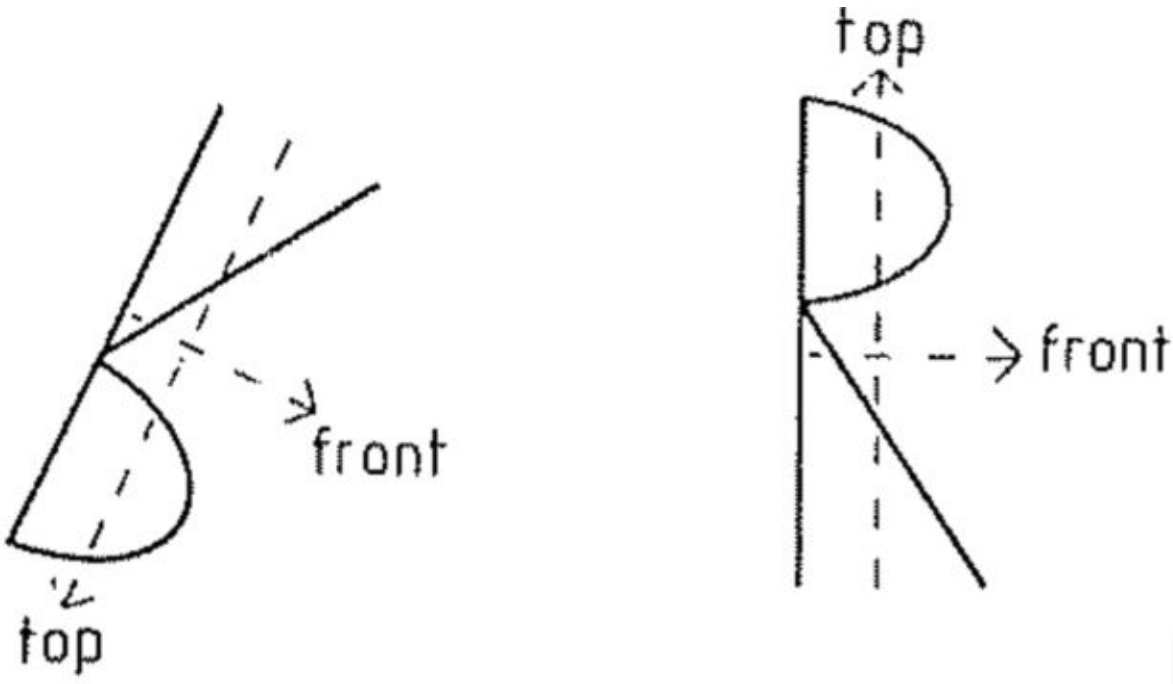
- **生物神经系统的思考。**
  - 反向传播难以成立。神经系统需要能够精准地求导数，对矩阵转置，利用链式法则，这种解剖学上从来也没有发现这样的系统存在的证据。
  - 神经系统含有分层，但是层数不高。且生物系统传导在ms量级（GPU在us量级），并且同步也出现问题



https://zhuanlan.zhihu.com/p/29435406

- 大部分哺乳类，特别是灵长类大脑皮层中大量存在称为 Cortical minicolumn 的柱状结构（皮层微柱），其内部含有上百个神经元，并存在内部分层。这意味着人脑中的一层并不是类似现在NN的一层，而是有复杂的内部结构。

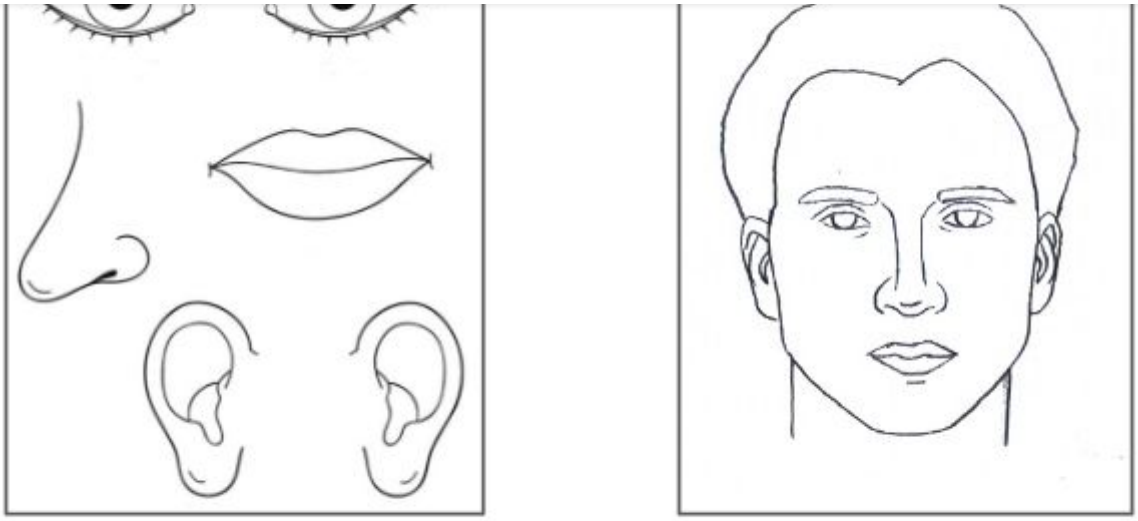
- **认知神经科学的思考**
  - 人会不自觉地会根据物体形状建立一种“坐标框架”(coordinate frame)。
  - 比如判断下面图字母是否一样。



- 我们需要通过旋转把坐标框架变得一致，才能从且宽上知道它们是否一致。

https://kndrck.co/posts/capsule_networks_explained/

  - Hinton认为：
    - **人的视觉系统会建立“坐标框架”，并且坐标框架的不同会极大地改变人的认知**
    - 人识别物体的时候，**坐标框架是参与到识别过程中**，识别过程受到了空间概念的支配。
    - 但是 **CNN没有“坐标框架”**
  - 但是在CNN上却很难看到类似“坐标框架”的东西。
  - Hinton 提出猜想：
    - 物体和观察者之间的关系（比如物体的姿态），应该由 **一整套激活的神经元表示** ，而不是由单个神经元，或者一组粗编码（coarse-coded，指类似一层中，并没有经过精细的组织）的神经元表示。
    - 这样的表示，才能有效表达关于“坐标框架”的先验知识。
- **CNN的目标不正确**
  - 先解释同变性（Equivariance）和不变性（Invariance）。
    - Invariance 不变性，物体表示不随变换变化。
      - 如空间的 Invariance，是对物体平移之类不敏感（物体不同的位置不影响它的识别)
    - Equivariance 同变性，用变换矩阵进行转换后，物体表示依旧不变.
      - 它是对物体内容的一种变换



https://kndrck.co/posts/capsule_networks_explained/

- CNN对旋转没有不变性。可采用 **数据增强方式**让其达到旋转不变性。
- CNN中的Invariance 主要是通过 Pooling 等下采样过程得到。而卷积层是同变性（Equivariance）。（相应已有措施？）
- 平移和旋转的Invariance，是舍弃了“坐标框架”。
- **虽然以往CNN的识别准确率高且稳定，但我们最终目标不是为了准确率，而是为了得到对内容的良好表示，从而达到“理解”内容。**

Hinton 提出的Capsules

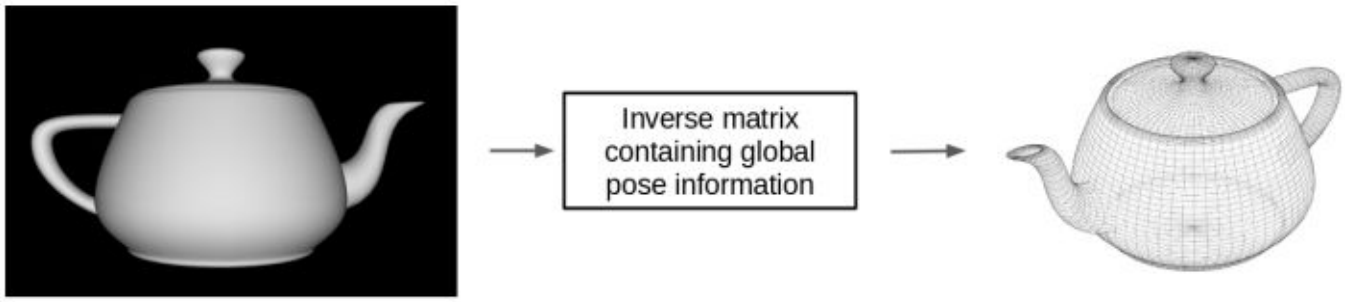
从上面介绍中得知Capsules需具备的性质有：

- 一层中有复杂的内部结构
- 能表达“坐标框架”
- 实现同变性（Equivariance）

Capsule用一组神经元而不是一个来代表一个实体，且仅代表一个实体。它是一个高维向量：

- **模长代表某个实体（某个物体，或者其一部分）出现的概率。**
- **方向/位置代表实体的一般姿态 (generalized pose)，包括位置，方向，尺寸，速度，颜色等等。**

知乎 首发于 SCUT神经网络-Magellan小分队



https://kndrck.co/posts/capsule_networks_explained/

Hinton认为存在两种同变性Equivariance：

- **位置编码**（place-coded）：视觉中的内容的位置发生了较大变化，则会由不同的 Capsule 表示其内容。
- **速率编码**（rate-coded）：视觉中的内容为位置发生了较小的变化，则会由相同的 Capsule 表示其内容，但是内容有所改变。
- 两者的联系是，高层的 capsule 有更广的域 (domain)，所以 **低层的 place-coded 信息到高层会变成 rate-coded。**

第一篇《Dynamic Routing Between Capsules》解析

- Dynamic Routing Between Capsules
- 先读懂CapsNet架构然后用TensorFlow实现：全面解析Hinton的提出的Capsule

Capsule 是一组神经元，其输入输出向量表示特定实体类型的实例化参数（即特定物体、概念实体等出现的概率与某些属性）。我们使用输入输出向量的长度表征实体存在的概率，向量的方向表示实例化参数（即实体的某些图形属性）。同一层级的 capsule 通过变换矩阵对更高级别的 capsule 的实例化参数进行预测。当多个预测一致时（本论文使用动态路由使预测一致），更高级别的 capsule 将变得活跃。

Capsule 是一个高维向量，模长表示概率，方向表示属性。故Hinton采用Squashing的非线性函数作为capsule的激活函数。

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

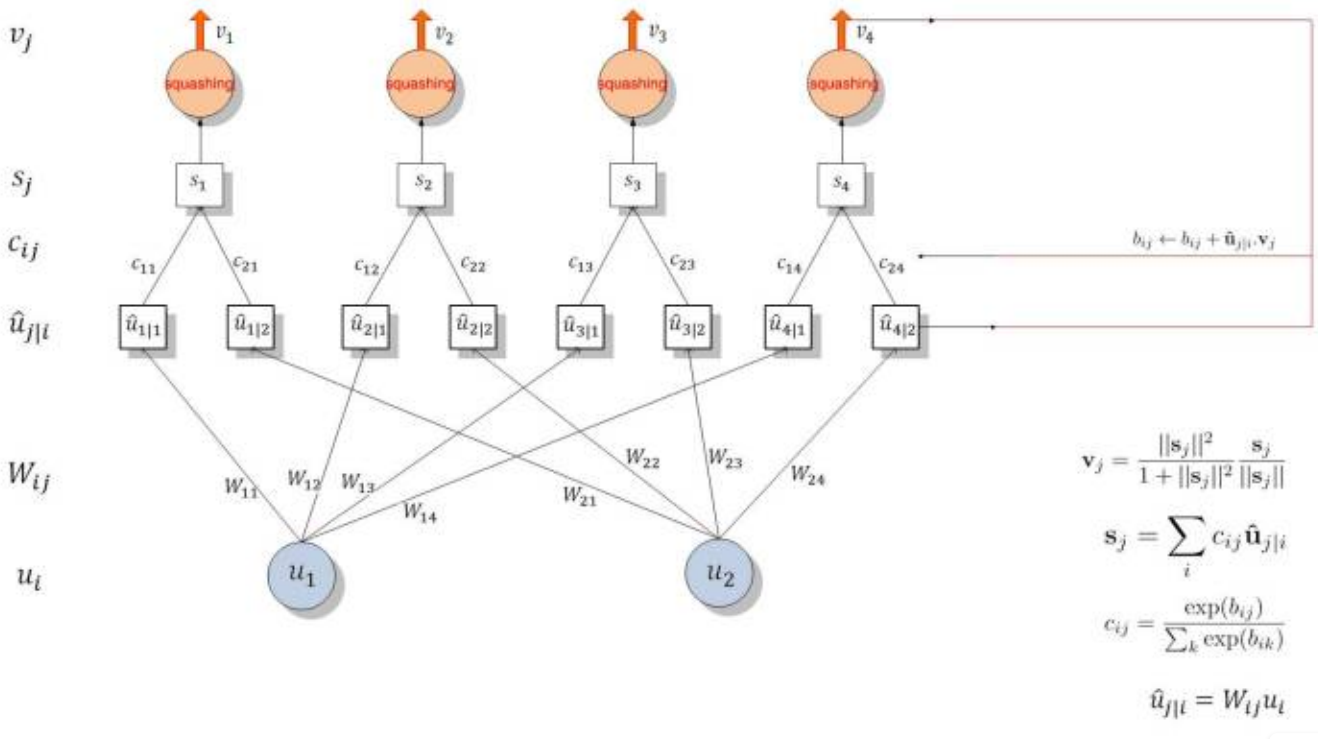
- 其中 v_j 为 Capsule j 的输出向量，s_j 为上一层所有 Capsule 输出到当前层 Capsule j 的向量加权和。 即 s_j 为 Capsule j 的输入向量。
- 该非线性函数前一部分是输入向量 s_j 的缩放尺度，后一部分是输入向量的单位向量s_j.

Capsule_j的输入向量 s_j的获取计算，是两层间的传播与联系方式。S_j计算过程分为两步，**线性组合 和 Routing**：

- 下图左式是Routing，右式是线性组合。

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \ , \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$$

- **Routing流程:**



https://www.jiqizhixin.com/articles/2017-11-05

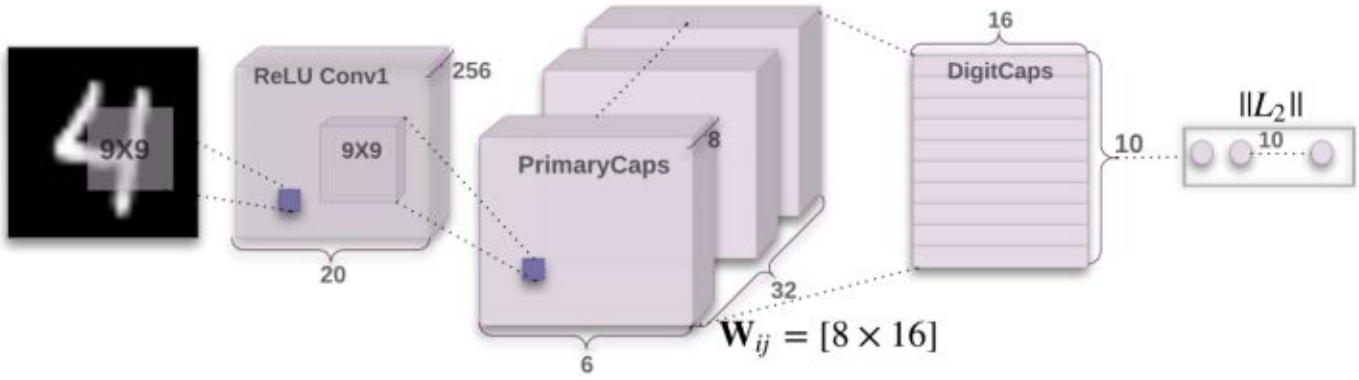
- s_1为u_1\1_hat和u_1\2_hat的线性组合而得出。而它们线性组合的系数大小则由c来决定。c_ij是常量。
- c的数值由b决定，而这个迭代运算中，b_ij 依赖于两个 Capsule 的位置与类型，但不依赖于当前的输入图像。比如u_1|2_hat 更接近v_1，那对应的b_12在下次迭代会增大，从而c_12大于c_11，从而让它更接近v_1。b_ij是常量。

算法总流程：

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ softmax computes Eq. 3
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ squash computes Eq. 1
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

CapsulesNet架构图：



- 第一个卷积层使用 256 个 9×9 卷积核，深度为1，步幅为1，且使用了 ReLU 激活函数。输出的张量20×20×256。此外，CapsNet 的卷积核感受野使用的是 9×9。这两层间的权值数量应该为 9×9×1×256+256=20992，后面加256是偏置值。
- 第二个卷积层开始作为 Capsule 层的输入而构建相应的张量结构。使用32×8个9×9卷积核（深度为256），步幅为2。输出张量为6×6×8×32，即**输出了6×6×32个维度为8的capsule向量**。两层间的权值数量为9×9×256×8×32+8×32=5308672。

- 第三层DigitCaps在第二层输出的向量基础上进行传播与Routing更新。第二层共输出6×6×32=1152 个capsule，即i层共有1152个Capsule，第三层j层有10个Capsules（每个是16维的向量）。
  - W_ij有1152×10个，每个是8×16的向量
  - u_i（8×1的向量）与W_ij相乘得到预测向量([8,16].T*[8,1]=[16,1])后，接着就有1152×10个耦合系数c_ij。
  - 将s_j传入squashing后就得到激活后的最终输出v_j。
  - DigitCaps 层与 PrimaryCaps 层之间的参数中，所有 W_ij 的参数数量是1152×10×8×16=1474560，c_ij 的参数数量为 1152×10，b_ij参数数量是1152×10。

损失函数和最优化：

耦合系数 c_ij 是通过一致性 Routing 进行更新的，但是**整个网络其它的卷积参数和 Capsule 内的 W_ij 都需要根据损失函数进行反向更新**。

作者采用了 SVM 中常用的 Margin loss，该损失函数的表达式为：

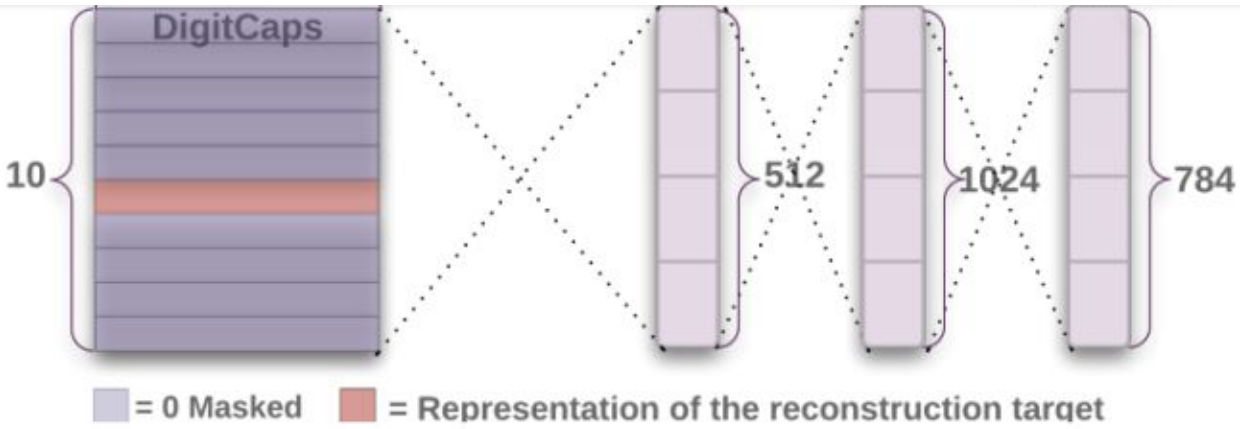
$$L_c = T_c \max(0, m^+ - ||\mathbf{v}_c||)^2 + \lambda (1 - T_c) \max(0, ||\mathbf{v}_c|| - m^-)^2$$

- 其中 c 是分类类别，T_c 为分类的指示函数（c 存在为 1，c 不存在为 0），m+ 为上边界，m- 为下边界。此外，v_c 的模即向量的 L2 距离。
- 对每一个表征数字 k 的 Capsule 分别给出单独的 Margin loss。
- 实例化向量的长度来表示 Capsule 要表征的实体是否存在。

重构与表征

利用预测出来的Capsules，重新构建出该类别的图像

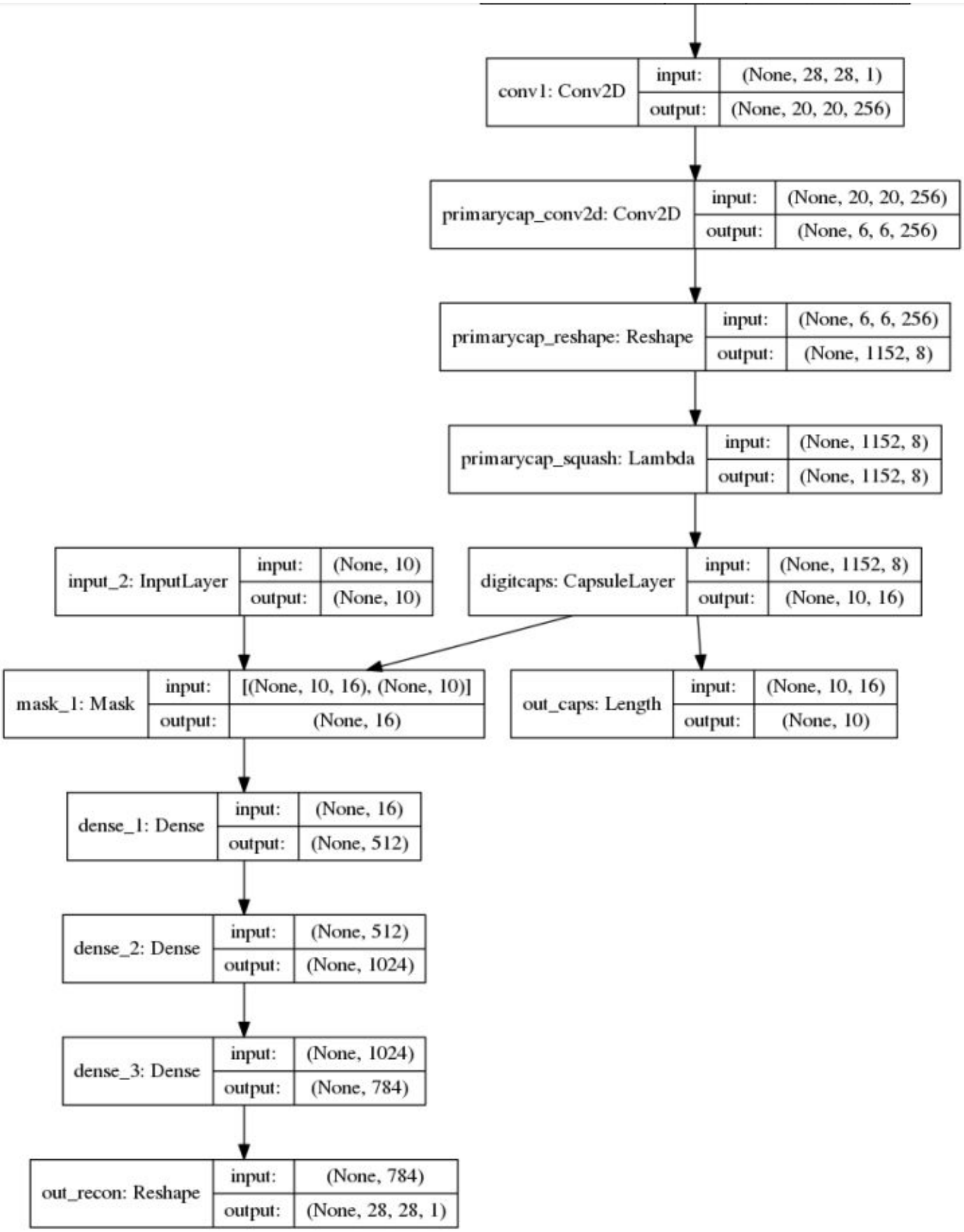
文章中使用额外的重构损失（reconstruction loss）来促进 DigitCaps 层对输入数字图片进行编码：



原图片可能数字重叠覆盖，导致预测出来的10个Capsules中部分Capsules的模都很大，所以重构单个数字图片时候，需要将其他Capsule进行Mask遮住，让有代表的Capsules去重构图片。损失函数通过计算在最后的 FC Sigmoid 层采用的输出像素点与原始图像像素点间的欧几里德距离作为损失函数。

为防止重构损失主导了整体损失（从而体现不出Margin loss作用），作者还按 0.0005 的比例缩小重构损失。

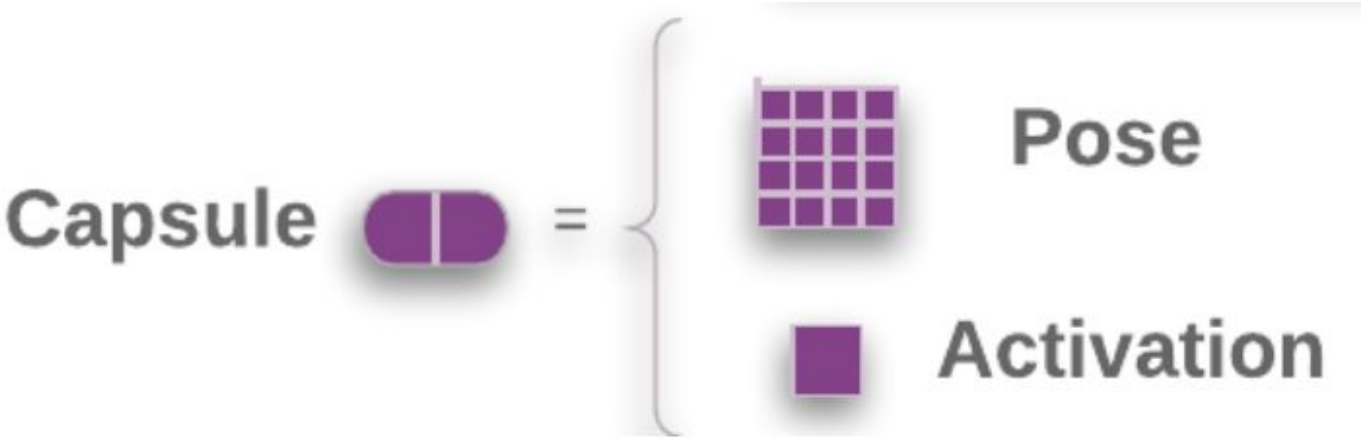
相比Softmax，Capsules不受多类别重叠的干扰，非常适合做单样本预测多类别的工作。



知乎 首发于 SCUT神经网络-Magellan小分队

Matrix Capsules with EM routing

Capsule结构:



- 4×4的姿势矩阵和1的激活值
- L层的某Capsule_i通过将自身的**Pose矩阵与视角不变矩阵的变换矩阵（viewpoint-invariant transformation matrix)相乘**，从而**为L+1层的许多不同Capsule的Pose矩阵进行投票**。 这些投票都会根据分配的**系数加权**。而这些系数进行**EM算法迭代更新**。

Matrix CapsuleNet的模型结构：

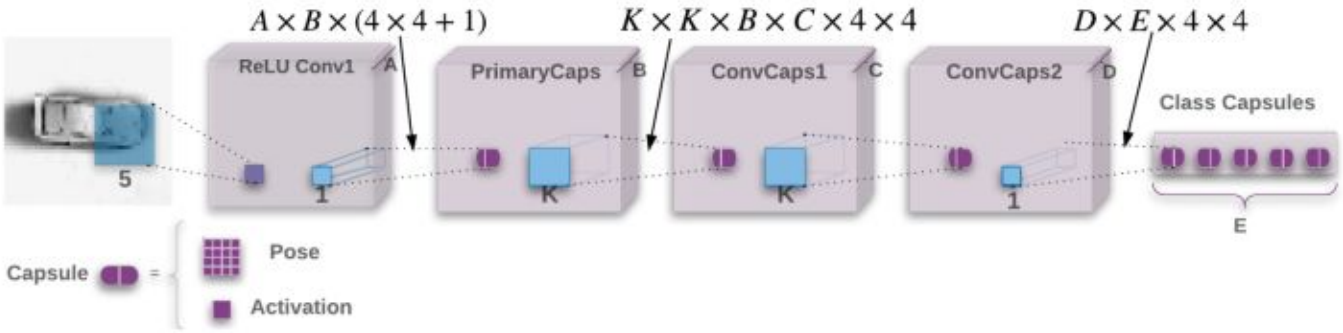


Figure 1: A network with one ReLU convolutional layer followed by a primary convolutional capsule layer and two more convolutional capsule layers.

Procedure 1 Routing algorithm[1] returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ich} is an H dimensional vote from capsule i with activation a_i in layer L to capsule c in layer $L+1$. β_a, β_v are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

procedure EM ROUTING($\boldsymbol{a}, V$)
  $\forall i, c$: $R_{ic} \leftarrow 1/size(L+1)$
  for t iterations **do**
    $\forall c$: $M_{c:}, S_{c:}, \boldsymbol{a}'_c \leftarrow$ M-STEP($R_{:c}, \boldsymbol{a}, V_{:c:}$)
    $\forall i$: $R_{i:} \leftarrow$ E-STEP($M, S, \boldsymbol{a}', V_{i::}$)
  return $\boldsymbol{a}', M$

procedure M-STEP($\boldsymbol{r}, \boldsymbol{a}, V'$) ▷ for one higher-level capsule
  $\forall i$: $\boldsymbol{r}'_i \leftarrow \boldsymbol{r}_i * \boldsymbol{a}_i$
  $\forall h$: $\boldsymbol{\mu}_h \leftarrow \frac{\sum_i \boldsymbol{r}'_i V'_{ih}}{\sum_i \boldsymbol{r}'_i}$
  $\forall h$: $\boldsymbol{\sigma}_h^2 \leftarrow \frac{\sum_i \boldsymbol{r}'_i (V'_{ih} - \boldsymbol{\mu}_h)^2}{\sum_i \boldsymbol{r}'_i}$
  $cost_h \leftarrow (\beta_v + log(\boldsymbol{\sigma}_h)) \sum_i \boldsymbol{r}'_i$
  $a' \leftarrow sigmoid(\lambda(\beta_a - \sum_h cost_h))$
  return $\boldsymbol{\mu}, \boldsymbol{\sigma}, a'$

procedure E-STEP($\boldsymbol{a}', S, M, V''$) ▷ for one lower-level capsule
  $\forall c$: $\boldsymbol{p}_c \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi \boldsymbol{S}_{ch}^2}} e^{-\sum_h^H \frac{(V''_{ch} - \boldsymbol{M}_{ch})^2}{2\boldsymbol{S}_{ch}^2}}$
  $\forall c$: $\boldsymbol{r}_c \leftarrow \frac{\boldsymbol{a}'_c \boldsymbol{p}_c}{\sum_j \boldsymbol{a}'_j \boldsymbol{p}_j}$
  return $\boldsymbol{r}$

参数介绍：

下标 i,c,h：

- i: 指L层中的某个capsule。
- c: 指L+1层中的某个capsule。
- h: 指Pose矩阵中的某维，共16维。

Pose矩阵：

- capsule的姿态矩阵，向量，此处是4×4的矩阵（或16×1的向量），方向表示属性。
- L层的某Capsule_i通过将自身的**Pose矩阵与视角不变矩阵的变换矩阵（viewpoint-invar transformation matrix)相乘**，从而为L+1层的许多不同Capsule的Pose矩阵进行投票。

视角不变的变换矩阵W：

- 在图形学中，某图片乘上W后，能够得到一定姿态旋转的图片。
- W和第一篇中的W是一样的，都是通过反向传播更新。

投票矩阵V

- V_ic= i_pose*W，当i的pose乘上W，得到i旋转变换后的姿态，该新姿态作为给c的投票（因为c的姿态是很多个i的姿态投票加权和，加权由概率p和激活值a等）。
- 很多个i意思是：如第一次capsule卷积层中，L有14×14×32个capsule_i，L+1有6×6×32个capsule_c.那卷积核大小为3×3×32（深度32），那此时很多i是指有3×3×32个i。在最后卷积层平拉转换成E个capsules时候，这很多个i指的是1×1×32 个capsule_i。

激活值 a、a_hat：

- capsule的激活值a，标量。表示当前capsule被激活的数值(0,1).
- 通过sigmoid函数激活.
- a表示L层的capsule_i的激活值.
- a_hat表示L+1层的capsule_c的激活值.

投票的加权系数R：

- 初始化为R_ic = 1/size(L+1), 其中size(L+1)为L+1层的capsule个数或者为3×3×32（即卷积核的大小和深度）。
- R_ic表示capsule_i投票给capsule_c的概率。
- r'_i表示前面所有层汇总后通过i投票给c的概率

算法中的M，S：

- M_ch: 指L+1层中capsule_c的Pose矩阵中h维的数值。M_c即为L+1层的Pose矩阵。
- S_ch^2: 指L层中capsules投给L+1层中capsule_c的Pose矩阵中h维数值的方差。

高斯概率P

- page3_(1)中提到的P_ich是单高斯模型。
  - i是样本，h和c是定值。
  - 样本i_h有size(i)个（数值取决与卷积核大小，如primaryCaps到ConvCaps1中就有3×3×32个样本）
  - 对于某变量i，传入P_ich中，能得到 **i中的h维的数值，属于c中h维的数值的相似度或说概率**
- 在E步中的p_c，即P_ic是混合高斯模型。
  - 将i的pose传入，得到i与c的相似度或说i属于c类的概率值.
  - 即i和c在每个h维的高斯混合计算得出，(每维的相似概率值)^h=i输入c的概率。
  - **p_c表示i给c的投票概率，而r_c就表示前面所有层汇合到L层并通过i投票给c的概率。**
- 考虑到转换矩阵，V=pose×W，让 **V_ic表示i的pose矩阵**，代入上面说法即可。

卷积层的理解：

primaryCaps到ConvCaps1的卷积层：

即L层有14*14*32个capsules，L+1层有6*6*32个capsules。 文章中的混合高斯模型，是指:

- L+1层的每个c都有一个高斯混合模型，其中k=h=16,即c有一个k=16高斯混合模型。
- 每个i样本（共3x3x32=288个样本），通过高斯混合模型，逐次对h维进行预测，最终相乘得到样本i属于c的概率（相似度）。
- 其中这高斯混合模型中，单个高斯是指i中的h（若不乘上转换矩阵）属于c中的h类的概率（此概率即为公式中的p_h)，将p_ich在i（288个）上叠加起来（再乘上a_hat)，就可以得到i属于c类的概率。

ConvCaps1到Class Capsules的卷积层：

最后一层是E个capsules，这是一个拉伸后的扁长Capsule层。

- 拉伸后的扁长层无法表达每个capsule的位置信息（前面的层的每个capsule相对map都有对应的位置信息）。所以为了保持capsule的位置信息，作者就将最后层中的capsule的pose矩阵中前两个元素代表位置信息。通过让前一层的部分i将其i自身所处的位置（行列两个值，再缩放到一定比例，如缩放到(0,1)间)加到它的投票矩阵V_ic的前两个元素中，从而让c中pose的前元素能代表位置信息。

- 同一个类别姿态的但不同位置的capsule共享转换矩阵W，也就是同一个map中的6×6=32个capsule共享转换矩阵。其实做法比前面简单点，起码W缩小到32个转换矩阵了。

损失函数：传播损失(Spread Loss):

为了使训练过程对模型的初始化以及超参的设定 没那么敏感，作者采用传播损失函数来最大化 被激活的目标类与被激活的非目标类 之间的间距。a_t表示target的激活值，a_i表示Class_Capsules中除t外第i个的激活值。

$$L_i = (max(0, m - (a_t - a_i))^2, \quad L = \sum_{i \neq t} L_i$$

将m从0.2的小幅度开始，在训练期间将其线性增加到0.9，从而避免无用胶囊的存在。

那m是在训练过程中让m逐渐增大吗？为什么要这样做呢？

我的理解是：

- 当模型训练得挺好时候（中后期），每个激活值a_i的比较小而a_t比较大。此时m需要设置为接近1的数值。这很好理解。
- 当模型初步训练时候，很多capsules起的作用不大。激活值a_i和a_t相差不大，若此时m采用较大值如0.9,就会掩盖了(a_t-a_i)的作用，让0.9起主导作用。比如更新前的参数W1和更新后的参数W2,若m采用0.9,，则无论W1还是W2,获得的L都差不多，这会让W和capsules很为难。而设置m为较小值就能让（a_t-a_i）起到较好的作用。

模型结构分析（我的代码思路）：

image-->Relu_conv1

传入图片。对原图片进行普通卷积操作。

- 原图片input=[32,32,1]
- 卷积核大小5×5
- 卷积核个数32
- stride=2
- output=[14,14,32]

Relu_conv1->PrimaryCaps

一个Caps有17维，其中16维是4×4的pose，1维是activation。

- input=[14,14,32]
- 双分支卷积Pose分支：
  - input=[14,14,32]
  - 卷积核大小1×1
  - 卷积核个数32
  - （其实已经没有卷积操作了，卷积核大小和个数这个信息只是来辅助理解EMRouting的。这往下忽略所有的**卷积核个数 和 卷积核大小** 这个概念吧，我不太好描述了）
  - stride=1
  - output=[14,14,32,16]
    - output_reshape=[196,32,16]

- 双分支卷积Activate分支：
  - input=[14,14,32]
  - 卷积核大小1×1
  - 卷积核个数＃32
  - stride=1
  - output=[14,14,32,1]
    - output_reshape=[196,32,1]
    - output_reshape[i]表示同种位置i但很多姿势的activate
    - output_reshape[i][j]表示i位置的j姿势的activate
- 分支合并
  - input=[196,32,16],[196,32,1]
  - output=[196,32,17]
    - output[i,j,:16]指位置i在j姿势的pose
    - output[i,j,16]指位置i在j姿势的activate

PrimaryCaps->ConvCaps1

用3x3的Caps卷积对PrimaryCaps的map进行操作 PrimaryCaps的一个map有196个Caps，一个位置有32个Caps

- input=[196,32,17]
- split:
  - input_pose=[196,32,16]
  - input_activation=[196,32,1]
- 通过V=Pose*W 得到V值
  - Pose=[196,32,16]
  - W=[6272,1152,16]
  - 通过**部分的Pose**和W相乘得到V_c。遍历c时将V_c循环相加，之后得到V
  - V=[6272,1152,16]
- Capsules卷积：
  - 卷积核大小3×3
  - 卷积核个数＃32
  - stride=2
  - EM算法：
    - 按着公式进行运算即可得到M和a_hat

- outputs=[36,32,17]

ConvCaps1->ConvCaps2

- input=[36,32,17]
- split:
  - input_pose=[36,32,16]
  - input_activation=[36,32,1]
- 通过V_c=部分i_Pose*W，遍历c后得到V。
  - Pose=[36,32,16]
  - W=[1152,512,16]
  - V=[1152,512,16]
- Capsules卷积：
  - 卷积核大小3×3
  - 卷积核个数＃32
  - stride=1
  - EM算法：
    - 按着公式进行运算即可得到M和a_hat
    - 即L+1层的pose矩阵和激活值。M=[16,32,16],a_hat=[16,32,1]
- output_merge：
  - outputs=[16,32,17]

ConvCaps2-> Class_Capsules

- input=[16,32,17]
- split:
  - input_pose=[16,32,16]
  - input_activation=[16,32,1]
- 全连接层：
  - 在同一种类型的不同位置上共享转换矩阵参数（即同一张map上的capsule共享视角不变转换矩阵）：

- 需要得到V=[512,10,16]
- 获取V的做法：
  - 将W扩充为冗余矩阵[16,32,10,16]=>[512,10,16]
  - 此时卷积核为1×1深度为32，故将1×1×32个Pose_i乘上W得到V_ic。
  - 然后将Pose_i所处map的位置的行列缩放到(0,1),分别加入到V_ic的前两维h=0和1，进行**更新V_ic**.此时的V_ic表示一个位置上给c投票的向量值(这个位置i有32个capsules)。
  - 对ConvCaps2层（L层）每个位置循环下，就可以得到L层对c的投票值
  - 再对c循环下，就可以得到完整的V了。（或者这两个循环换下位置）
- EM算法：
  - 有了a和V，就可以Routing了。
  - 输出M=[10,16],a_hat=[10,1]
- output= [10,17] 输出10个Capsules

编辑于 2017-11-14

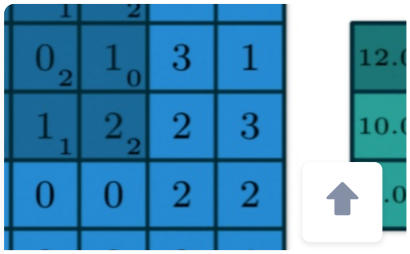
深度学习（Deep Learning） 卷积神经网络（CNN）

文章被以下专栏收录

推荐阅读

hinton capsule设计思想美感

本文主要讲2个问题：1，传统神经网络存在疑点的地方；2，capsule的解决之道； 一，传统cnn存在疑点的地方： 1，BP是一种数学技



知乎 首发于 SCUT神经网络-Magellan小分队

论智 川流不息 发表于AI算法背... 论智 发表于

29 条评论 切换为时间排序

写下你的评论...

吐司司司机 1 年前

请问一下，PrimaryCaps->ConvCaps1里面196是怎么到36的啊？

赞

Nango 明楠 (作者) 回复 **吐司司司机** 1 年前

L层中每3×3×32个i，投票浓缩到L+1层中的1×32个c。也就是L层中一个14×14map，变成1个6×6的map。注意，stride=2。

赞

Nango 明楠 (作者) 1 年前

更新了第二篇的完整思路。

赞

胡杨 1 年前

师弟的总结，比较全，参考的其他文章已经引用，还有一些师弟自己的想法

赞

程序媛 1 年前

PromaryCaps层capsule个数为14x14x32;
ConvCaps层capsule个数为6x6x32;作者应该是笔误少写了乘号

赞

Nango 明楠 (作者) 回复 **程序媛** 1 年前

对对，谢谢阿

赞

川流不息 1 年前

高斯概率P 这节很难理解,对混合高斯模型不熟悉:(有个形象点的图来说明就好了：

赞

我其实尝试画了下图，觉得太丑了，怕玷污了这篇论文 233333

1

我叫枫木木 1 年前

问下最后输出10个16*1的向量是吗，怎么做分类啊，是这10个向量代表10个数字吗，然后比较哪个向量值大就属于哪一类是吗

赞

Nango 明楠 (作者) 回复 **我叫枫木木** 1 年前

对，谁的模最长就认为属于哪一类。模长表示类在图片中的概率。适合做多类别共存的识别。

赞

我叫枫木木 回复 **Nango 明楠** (作者) 1 年前

那如果一张图片里面有两个数字，怎么分类呢，是设了一个阈值吗，概率值总是有个最大的吧，

赞

查看全部 7 条回复

susht 1 年前

答主，在Routing流程中你写：

c的数值由b决定，而这个迭代运算中，b_ij 依赖于两个 Capsule 的位置与类型，但不依赖于当前的输入图像。比如u_1|2_hat 更接近v_1，那对应的b_12在下次迭代会增大，从而c_12大于c_11，从而让它更接近v_1。b_ij是常量。

我觉得 u_1|2_hat 更接近v_1说明，u2对应的capsule更接近v_1，应该是 c21 > c22 c23 c24 吧 哪来的c_12和c_11。。

1

Nango 明楠 (作者) 回复 **susht** 1 年前

对，我文章中在这里解释错了。你的才是正确的，感谢指出。= =

赞

不去新一 11 个月

学习，感谢。

知乎 首发于 SCUT神经网络-Magellan小分队

哇噻 11 个月前

那个位置编码和速率编码是意味着比如一张脸进行平移或旋转，底层的位置编码发生了很大的变化，但是上层还是检测到一致性，只是上层的速率编码有所变化？

赞

指间少年 10 个月前

请问一下，在matrix capsule里面，变换矩阵W的大小是多少？这里和论文中的不太一样啊？一直没理解

赞

Nango 明楠 (作者) 回复 指间少年 9 个月前

我写下这文章后作者更新了论文并重新放出来,不过之后没时间去看 = =, 关于参数的维度可能有会误,希望能帮助到你.

赞

周大大 10 个月前

我想问一下，这个网络分为解码器和编码器，解码器和编码器是单独训练还是共同训练，是解码器将数字编码成16维向量先训练，然后得到16维向量输入到解码器中在进行训练，解码器和编码器互补影响，还是解码器和编码器是一个完整的网络，解码器的损失函数得到后再进行反向传播时会影响编码器的权值的权重

赞

ranjiewen 9 个月前

“Capsules不受多类别重叠的干扰，非常适合做单样本预测多类别的工作”，想请教一下，Capsules不受多类别重叠的干扰 具体体现在损失函数中吗？如果存在多类别重叠的现象，我们一般怎么处理？怎么设计损失函数？谢谢啦

赞

Nango 明楠 (作者) 回复 ranjiewen 9 个月前

确实是在损失函数里面.如果多类别重叠一般应该采用k(或k+1)个二分类器(共k类). 你可以搜下关于softmax的缺点分析.

赞

ranjiewen 回复 Nango 明楠 (作者) 9 个月前

谢谢啦，也可以使用代价敏感矩阵，使代价最小吧

赞

没头脑 6 个月

不变性和同变性有啥区别？文中写的是：都是不随变换而变化，是我理解错了吗？作者能进解

知乎 首发于 SCUT神经网络-Magellan小分队

Nango 明楠 (作者) 回复 没头脑 6 个月前

我理解是： 比如照片上有一手机（想象成平放在桌子上的手机，桌子边缘为图片边框）。如果将手机翻转下（比如立起来），那只具有不变性（Invariance）的系统并不能识别出它是手机。

赞

Ana 25 天前

可不可以简单介绍下转换矩阵的作用呢？不是很理解，是把不同角度的图像转正吗？可是固定的转换矩阵是怎么对不同角度的图像起作用的呢？

赞