Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Прикладной Математики и Информатики
Кафедра корпоративных информационных систем

Направление подготовки / специальность: 01.03.02 Прикладная математика и информатика

Направленность (профиль) подготовки: Прикладная математика и компьютерные науки

ИССЛЕДОВАНИЕ ИНФОРМАЦИОННОГО БЫТЫЛОЧНОГО ГОРЛЫШКА В НЕЙРОННЫХ СЕТЯХ

(бакалаврская работа)

Студент:

Николаев Михаил Алексеевич

_____

(подпись студента)

Научный руководитель:

Леонидов Андрей Владимирович

_____

(подпись научного руководителя)

Москва 2022

Аннотация

Данная работа посвящена изучению теории информации в нейронных сетях и исследованию Bottleneck Principe (принцип узкого места). В ходе работы была изучена взаимосвязь информации с алгоритмами машинного обучения на примере обучения нейронной сети классификации цифр по картинке.

Содержание

1. Проблема

1.1. Введение

Представим, что нам требуется обучить нейронную сеть. Будем рассматривать обучение с учителем. Тогда у нас есть вход X и выход Y, во время обучения нейронной сети алгоритм пытается найти оптимальные веса, чтобы по входу $x_i \in X$ сеть получала выход $y_i \in Y$. Нейронная сеть представляет собой последовательность слоев $\{Z_0, Z_1, Z_2, ..., Z_n\}$ где $Z_0 = X, Z_n = Y^{'}$,где $Y^{'}$ предсказание сети, и набор функций преобразований $\{f_1, f_2, ..., f_n\}$ и веса $\{\theta_1, \theta_2, ..., \theta_n\}$, где $Z_{i+1} = f_i(Z_i, \theta_i)$. Соответственно сеть преобразовывает исходное пространство X в латентное представление Z, которое является сжатым представлением пространства X. Это значит что у хорошей модели латентное пространство включает в себя всю необходимую информацию о входе для представления предсказания о выходе. То есть нам необходимо перенести из входа минимальное количество информации для латентного пространства для максимально подробного описания выхода. Эта оптимизационная задача называется принципом бутылочного горлышка. Хочется решить эту оптимизационную задачу и изучить как перетекает информация между слоями взависимости от качества сети.

1.2. Формальная постановка задачи

1. Изучить литературу про теорию информацию в нейронных сетях, взять за основу данные о поведении информарционных метрик в течение обучении сети и использовать их как гипотезу.

2. Взятие за основу легкую нейронnуй сеть (сеть классификации цифр по фото), построить алгоритм ее обучения.

3. Подробно изучить логическое устройство каждого из пространсв, на основе этих знаний построить статистическую оценку для кажого слоя.

4. Построение на ней множества вероятностных пространств - для каждого слой свое множество пространств. Каждое вероятное пространство задается гиперпараметрами.

5. Построить распределение для каждого слоя, и совместные распределения между слоями.

6. Посчитать информацинные метрики (такие как энтропия) на полученных распределениях.

7. Посмотреть на полученные информационные метрики в зависимости от эпохи обучения, сравнить с гипотезами, полученными с помощью литературы

8. Провести валидацию, на этапе которой подобрать наилучшие гиперпараметры вероятностных пространств и соответственно понять преимущества и недостатки каждого из вероятностных пространств.

9. Сделать выводы о проведенных экспериментов.

2. Обзор литературы

2.1. Базовые определения

Рассотротрим вероятностное просранство (X, Θ, p), где X просранство, Θ сигма- алгебра на X, p вероятносная мера введеная на сигма-алгебре Θ. Тогда введем энтропию

$$H[X] = -\mathbb{E}_{p(x)} \log(p(x)) \tag{1}$$

В дискретном случае

$$H[X] = -\sum_{x \in X} p(x) \log(p(x)) \tag{2}$$

Введем совместную энтропию

$$H[X, Y] = -\mathbb{E}_{p(x,y)} \log(p(x, y)) \tag{3}$$

В дискретном случае

$$H[X, Y] = -\sum_{x \in X, y \in Y} p(x, y) \log(p(x, y)) \tag{4}$$

Теперь введем условную энтропию

$$H[X|Y] = -\mathbb{E}_{p(x,y)} \log(p(x|y)) = H[X, Y] - X[Y] \tag{5}$$

$$\begin{aligned}
\square\ H[X, Y] - X[Y] = -\mathbb{E}_{p(x,y)} \log(p(x, y)) + \mathbb{E}_{p(y)} \log(p(y)) = \\
-\int_{x,y} p(x, y) \log(p(x, y)) + \int_{y} p(y) \log(p(y)) = \\
-\int_{x,y} p(x, y) \log(p(x, y)) + \int_{y} \sum_{x} p(y, x) \log(p(y)) = \\
-\int_{x,y} p(x, y) \log(p(x, y)) + \int_{x,y} p(y, x) \log(p(y)) = \\
-\int_{x,y} p(x, y)(\log(p(x, y)) - \log(p(y))) = \\
-\int_{x,y} p(x, y)(\log(p(x, y)/p(y))) = -\int_{x,y} p(x, y) \log(p(x|y)) = H[X|Y]\ \square
\end{aligned} \tag{6}$$

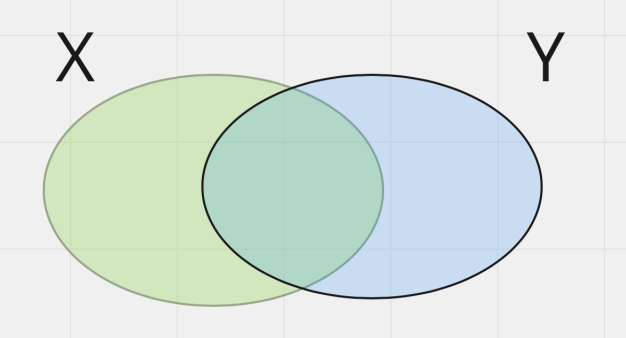
Теперь введем информацию между двумя величинами

$$I[X, Y] = -\mathbb{E}_{p(x,y)} \log(\frac{p(x)p(y)}{p(x, y)}) = H[X] + H[Y] - H[X, Y] \tag{7}$$

$$
\begin{aligned}
&\square\, H[X] + H[Y] - H[X,Y] = \\
-(\int_{x,y} p(y,x)\log(p(x)) + \int_{x,y} p(x,y)\log(p(y)) - \int_{x,y} p(y,x)\log(p(x,y))) &= \\
-\int_{x,y} p(y,x)(\log(p(x)) + \log(p(y)) - log(p(x,y))) &= \\
-\int_{x,y} p(x,y)\log(\frac{p(x)p(y)}{p(x,y)}) = I[X,Y]\,\square &
\end{aligned}
\tag{8}
$$

2.2. Информационные множества

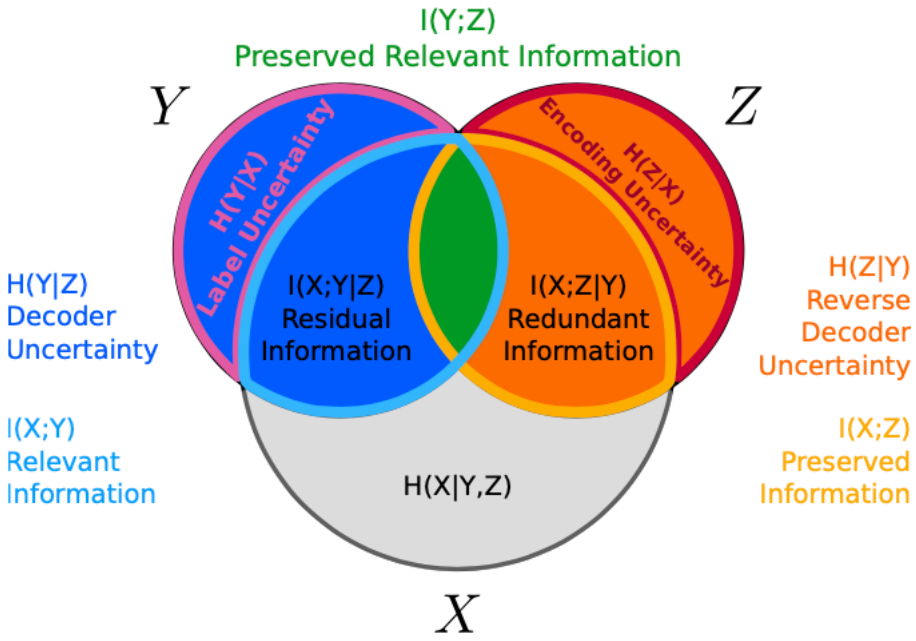
Таким образом информационные эвристики можно воспринимать как множественные операции



Рассмотрим пространства X и Y как множества, тогда мощностью множества

1. $|X|$ будем называть $H[X]$
2. $|Y|$ будем называть $H[Y]$
3. $|X \cap Y|$ будет $I[X,Y]$
4. $|X \cup Y|$ будет $H[X,Y]$
5. $|X \setminus Y|$ будет $H[X|Y]$
6. $|Y \setminus X|$ будет $H[Y|X]$

Перейдем к рассмотрению информационных множеств в рамках нейронных сетей. Тогда X рассмотрим как пространство входных данных, Y рассмотрим как пространство выходных данных, Z как латентное пространство. Визуализируем их в качестве множеств. Подобная диаграмма называется Mickey Mouse I-diagram.



Соответственно каждое из подмножеств численно вычисляется с помощью информационных метрик. Заметим, что $X \cap Y \cap Z = Y \cap Z$, иными словами $I[X, Y, Z] = I[Y, Z]$. Докажем это. $I[X, Y, Z] = I[Y, Z] \Leftrightarrow I[Y, Z] - I[X, Y, Z] = H[Y, Z|X] = 0$. Это верно, так как для каждого входа $x_i \in X$ зафиксирован конкретный выход $y_i \in Y$ и конкретное латентное кодирование $z_i \in Z$, а значит пространство $(Y, Z|X)$ будет одномерно для каждого $x_i \in X$, следовательно $H[Y, Z|X] = 0$.

2.3. Information Bottleneck Principe

Задача нейронной сети вычленить всю релевантную информацию из входа X описывающую выход Y. Оптимальное отображение будет отбрасывать всю нерелевантную информацию, таким образом задача обучения состоит в том, чтобы минимизировать общую информацию $I[X, Z]$, при этом сохраняя всю необходимую общую информацию между выходом и латентным пространством $I[Y, Z]$. Обощая эти факты, поиск оптимального решения описывается задачей оптимизации (IB):

$$min_Z I[X, Z] - \beta * I[Z, Y] \tag{9}$$

Где β оператор компромиса между информацией репрезентации $I[X, Z]$, и информацией релевантной информации $I[Z, Y]$. Посмотрев на Mickey Mouse I-diagram можно заметить, что множество Z пытается максимально влится в Y и иметь наименьшее пересечение с X. Проверим данные гипотезы на практике.

3. Архитектура нейронной сети

3.1. Описание задачи и датасета

В качестве примера возьмем нейронную сеть класификации цифр по картинке. За основу датасета возьмем MNIST датасет. В датасете педставлены чернобелые картинке рукописных цифр и соответсвующие им значения цифр, которые представлены на картинке. На каждом изображении изображена ровно одна цифра. Датасет разделен на 2 части: train и test для тестирования и обучения. Для практики выбрана такая задача, так как сеть для обучения получится легкой: в ней будет мало параметров, сеть обучается довольно быстро и за небольшое количество эпох, также данная задача хорошо подходит для исследования, так как на вход подается картинка, это пространство имеет размерность матрицы, на выходе же подается число, размерность которого одномерная. Соответственно в ходе преобразований сеть должна выявить все релевантные признаки и сильно сжать пространство, что полезно для нашего эксперимента.

3.2. Архитектура сети

Количество семплов в обучающей выборке 6000, размер каждого изображения (28, 28). Изображение одноканальное, так как оно чернобелое. Сеть предствляет собой следующую архитектуру

```
: CNN(
    (conv1): Sequential(
      (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
      (1): ReLU()
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (conv2): Sequential(
      (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
      (1): ReLU()
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (out): Linear(in_features=1568, out_features=10, bias=True)
  )
```

Подробная количественная таблица параметров сети:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 28, 28]             416
              ReLU-2           [-1, 16, 28, 28]               0
         MaxPool2d-3           [-1, 16, 14, 14]               0
            Conv2d-4           [-1, 32, 14, 14]          12,832
              ReLU-5           [-1, 32, 14, 14]               0
         MaxPool2d-6             [-1, 32, 7, 7]               0
            Linear-7                   [-1, 10]          15,690
================================================================
Total params: 28,938
Trainable params: 28,938
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.32
Params size (MB): 0.11
Estimated Total Size (MB): 0.44
----------------------------------------------------------------
```

Соответственно сеть начинается с 2 сверточных слоев (conv1, conv2). Свертки представляют собой последовательность: 2d-свертка, функция архивации (ReLU()) и пуллинг (MaxPool2d). Далее после последовательных применений сверток conv1 и conv2 к входному тензору, тензор транспонируется в двумерный тензор, далее к нему применяется полнозвязный слой и функция Softmax. Финальное представление должно предсказывать вероятностное распределение на цифрах.

3.3. Цикл обучения сети с подробным forward pass

1. $z_1 = conv1(x)$, где x - входной батч, conv1 - первая свертка. $x.shape = (batch\_size, 1, 28, 28)$
2. $z_2 = conv2(z_1)$, где z_1 - выход первого слоя сети, иначе говоря первого латентного пространства, conv2 - вторая свертка. $z_1.shape = (batch\_size, 16, 28/2, 28/2) = (batch\_size, 16, 14, 14)$.
3. $z_2^{flatten} = flatten(z_2)$, где z_2 - выход второго слоя сети, $flatten$ - функция меняющая размерность тензора, а именно превращая все признаки в одномерное векторное пространство. $z_2.shape = (batch\_size, 32, 14/2, 14/2) = (batch\_size, 32, 7, 7)$.
4. $z_3 = linear(z_2^{flatten})$, где $z_2^{flatten}$ - выход второго одномерного слоя сети, $linear$ - полносвязный слой. $z_2^{flatten}.shape = (batch\_size, 32 * 7 * 7) = (batch\_size, 1568)$.
5. $target = softmax(z_3)$, где z_3 - выход третьего слоя, функция softmax позволяет репрезентовать распределение, как вероятностное распределение в отрезке $[0, 1]$, $z_3.shape =$

$(batch\_size, 10)$

$$softmax(x_i) = \exp(x_i) \setminus (\sum_j \exp(x_j)) \tag{10}$$

6. Выход y представляется как one hot вектор y_{onehot}, например

$$y = 3 \Leftrightarrow y_{onehot} = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] \tag{11}$$

7. y_{onehot} сравнивается с $target$ с помощью лосс функции кросс энтропии L

$$L(y, t) = -\sum_i y_i * \log(t_i) \tag{12}$$

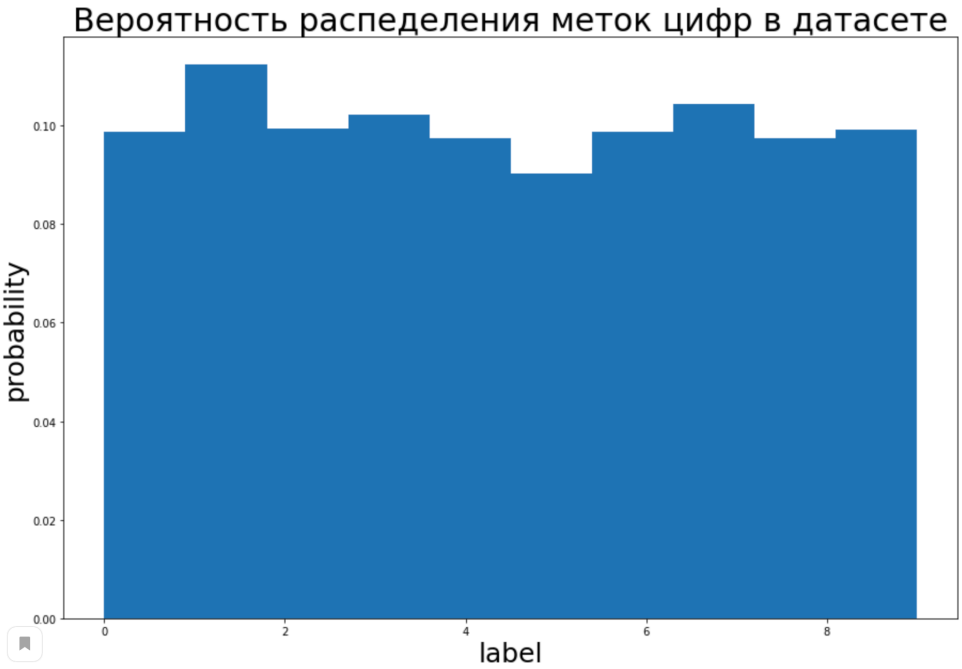
Данная сеть обучалась с помощью инфмационной метрики в лоссе - кросс энтропии между таргетом и ground truth. При этом не учитывались никакие другие информационные эвристики в других латентных пространствах (z_1, z_2, z_3).

4. Вероятностное пространство

Для работы с информационными эвристиками нам нужно построить вероятностное пространство для каждого слоя. Пространства входа X и выхода Y не изменяются в зависимости от эпохи обучения. При этом латентное пространство Z будет менятся с каждыми новыми весами моделей.

4.1. Вероятностное пространство на выходном слое

Проше всего задать вероятностное пространство на выходном слое, так как это категоричный признак, он принимает только целые числа в отрезке [0, 9]. Соотвенственно если данные в семплах распредены равномерно, то вероятность выпадания каждого класса - 0.1. проверим это посчитав количествено все метри на датасете.



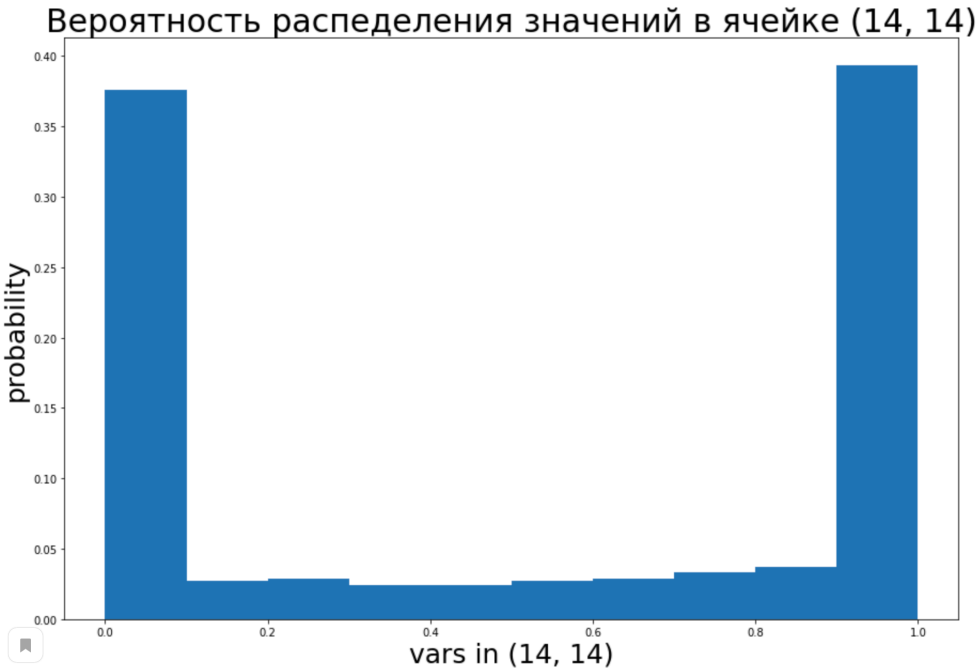
Как видно на графике распределение на выходном слое почти равномерное, будем считать что вероятность для каждого элемента будет 1 / 10. Сложнее ситуация с пространства входа и латентными, так как они многомерные.

4.2. Вероятностное пространство на входном слое

Сначала разберемся с входным пространством. Его размерность $(samples\_count, 1, 28, 28)$. Рассмотрим конкретную координату картинки и посмотрим как распределены значения на

конкретном признаке.

В качестве примера возьмем координаты (14, 14) и (7, 7):



Видно, что значения внутри каждой ячейки распределены нестандартно, привалирующие значения - 0 и 1. Также правильнее рассматривать модель, где значения в каждой ячейке зависят друг от друга, так как картинки являются не случайным шумом, а непрерывными матрицами, то есть если все значения вокруг ячейки равно 1., то скорее всего значение в ячейки тоже 1. Проверим этот факт независимости.

Возьмем квадратик из 4 ячеек и проверим вероятность условия, что все значения больше 0.5 и также посчитаем перемноженную вероятность что значение в каждой ячейке больше 0.5 и

сравним эти значения

$$P(x_{i,j} > 0,5, x_{i,j+1} > 0,5, x_{i+1,j} > 0,5, x_{i+,j+1} > 0,5)$$
$$? \tag{13}$$
$$P(x_{i,j} > 0,5), *P(x_{i,j+1} > 0,5) * P(x_{i+1,j} > 0,5) * P(x_{i+,j+1} > 0,5)$$

Рассмотрим конкретный пример, описанный выше со значениями i = 14, j = 14

```
i = 14
j = 14
condition1 = (input_x[:, 0, i, j] > 0.5) * 1.
condition2 = (input_x[:, 0, i, j + 1] > 0.5) * 1.
condition3 = (input_x[:, 0, i + 1, j] > 0.5) * 1.
condition4 = (input_x[:, 0, i + 1, j + 1] > 0.5) * 1.
prob_mult = (condition1.sum() / len(input_x)) * \
            (condition2.sum() / len(input_x)) * \
            (condition3.sum() / len(input_x)) * \
            (condition4.sum() / len(input_x))
prob_joint = (condition1 * condition2 * condition3 * condition4).sum() / len(input_x)
print("совместная вероятность события:            ", prob_joint)
print("перемножения вероятностей каждого события: ", prob_mult)
```

```
совместная вероятность события:             0.3325166666666667
перемножения вероятностей каждого события:  0.08890247455508297
```

Как видно на примере значения совершенно различные что говорит о зависимости между признаками входа.

Для удобства дискретизируем все значения входа, так как значения на входе принимают вещественные значения на отрезке [0, 1]. Четкое количество кусков разбиения оставим в качестве гиперпараметра и подберем при валидации. При дикретизации мы будем иметь дело с конечным пространством, что удобнее. Пример такой дискретизации:

$$X_{discrete} = (X_{input} > 0.5) * 1. \tag{14}$$

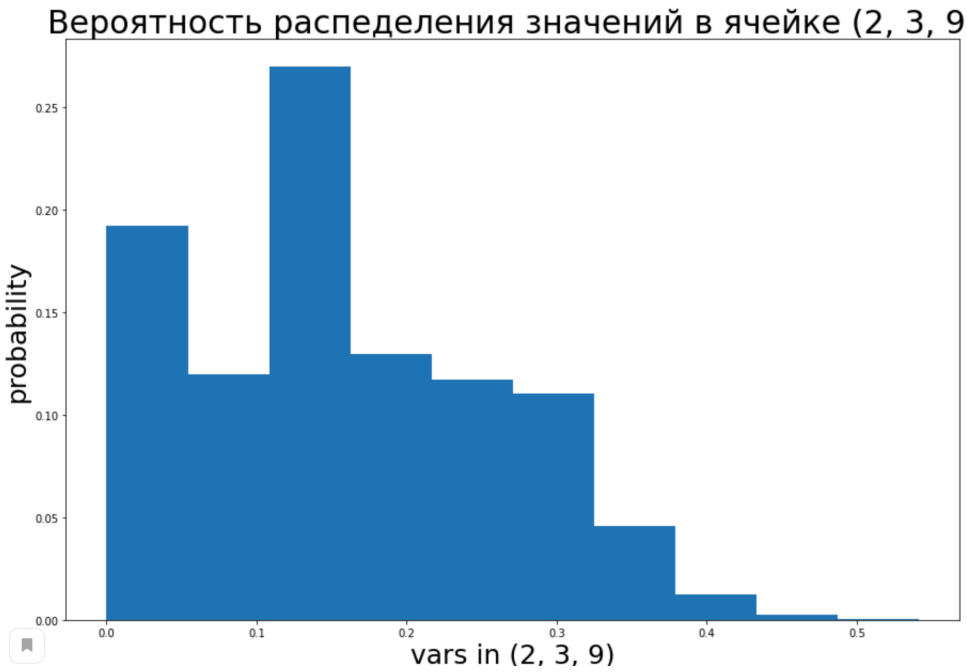
Данная дискретизация приводит все значения к двум полярным - 0 и 1, данная дикретизация нам подходит, так как на примерах распределения значений по признакам значения с наибольшими вероятностями были именно полярные значения.

На данный момент непонятно какими вероятнострыми моделями приближать такие сложные многомерные пространства как входное, поэтому наилучшим решением будет рассматривать исходы как элементы построенного дискретного пространства. Может быть проблема, что каждый элемент датасета будет инъективно отображаться в построенное дискретное пространства, для проверки этого посчитаем количество различных исходов. Пусть количество параметров при дискретизации будет минимально, а именно 2. То есть все входные значения переходят либо в 0 либо в 1. В таком случае полученное пространство будет иметь

2^{28*28} различных исходов, тем временем количество элементов в датасете 60000, что на много порядков меньше чем 2^{28*28}. Для решение этой проблемы используем искуственное сжатие пространства, которое рассмотрим чуть позже.

4.3. Вероятностное пространство на латентных пространствах

Для начала посмотрим как устроены данные на промежуточных слоях. Это также многомерные пространства, посмотрим на пример распределения одного из признаков слоя после первой свертки:



Данные здесь получаются после применения 2d свертки, ативации RELU и после пуллинга, соответственно все данные меньшие 0 стали равны 0, можно сделать вывод что после сверки данные имели нормальное распределение с шумом, но после функции активации стали иметь распределение как у функции RELU(norm), описать стандартной моделью подобные данные не получится.

Также данные будут зависимы друг с другом по призакам, так как значения в латентных слоях являются функцией от входных данных, которые выполняются на forward pass. Так как forward pass будет работать почти инвариантно на первых слоях сети из-за больших размеров пространств зависимость созраняется.

Соответственно для построения латентного пространства мы также проводим дискретизацию, чтобы привести непрерывное пространство в конечное. Для этого мы находим квантили распределения (например 0.01 и 0.99 квантиль) и ограничиваем нашу область по признаку этой квантилью. Значения будут находится за ее пределами с вероятностью $0.01+(1-0.99) =$

0.02. Далее разбиваем полученный отрезок на несколько равномерных кусков и дискретизируем. Количество кусков оставим в качестве гиперпараметра и подберем его при валидации. При расчете полученного пространства после дискретизации возникают те же проблемы, что и на входном пространстве. Если рассматривать в качестве исходов - элементы дискретизации, то распределение получится также равномерным в силу того, что количество семплов в датасете намного меньше чем возможных вариантов исходов, поэтому прибегнем к такому же сужению пространства, поговорим про него поподробней.

4.4. Сужение вероятностных пространств

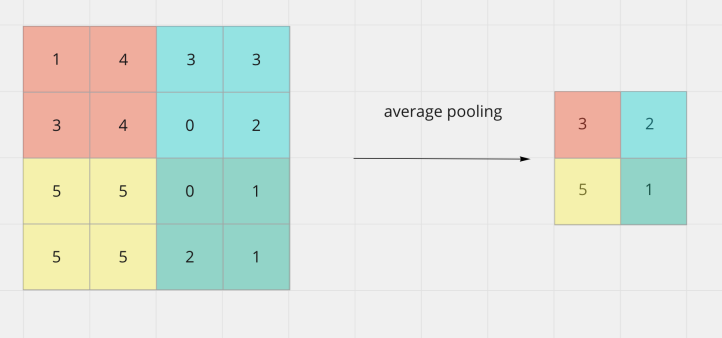
Как выяснилось в предъидущих пунктах, работать с вероятностным пространством без сужения будет бессмысленно, покажем это на практике в главе ниже.

Для снижения пространства будем использовать пуллинги, то есть объединение соседних признаков тензора в один. Эта операция сохраняет вероятно-признаковое пространство, так как в архитектурах сети часто используются пуллинги (как например в нашей сети классификации цифр по картинке). Особенности пуллинга также будем использовать как гиперпараметр.

Для двумерного пространсва такого как нащи исходные данные будем использовать $2daveragepooling$ Если использовать окно 7x7 то после пуллинга размер матрицы будет $(28/7) * (28/7) = 16$. В таком случае количество исходов полученного вероятностного пространства будет $2^{16} = 65538$, что уже сопоставляется с размерностью исходного датасета. Уже в этом случае

$$p(f(x_i) = f(x_j) \cap i \neq j) \ll 1, \qquad f = discrete \circ pooling \tag{15}$$

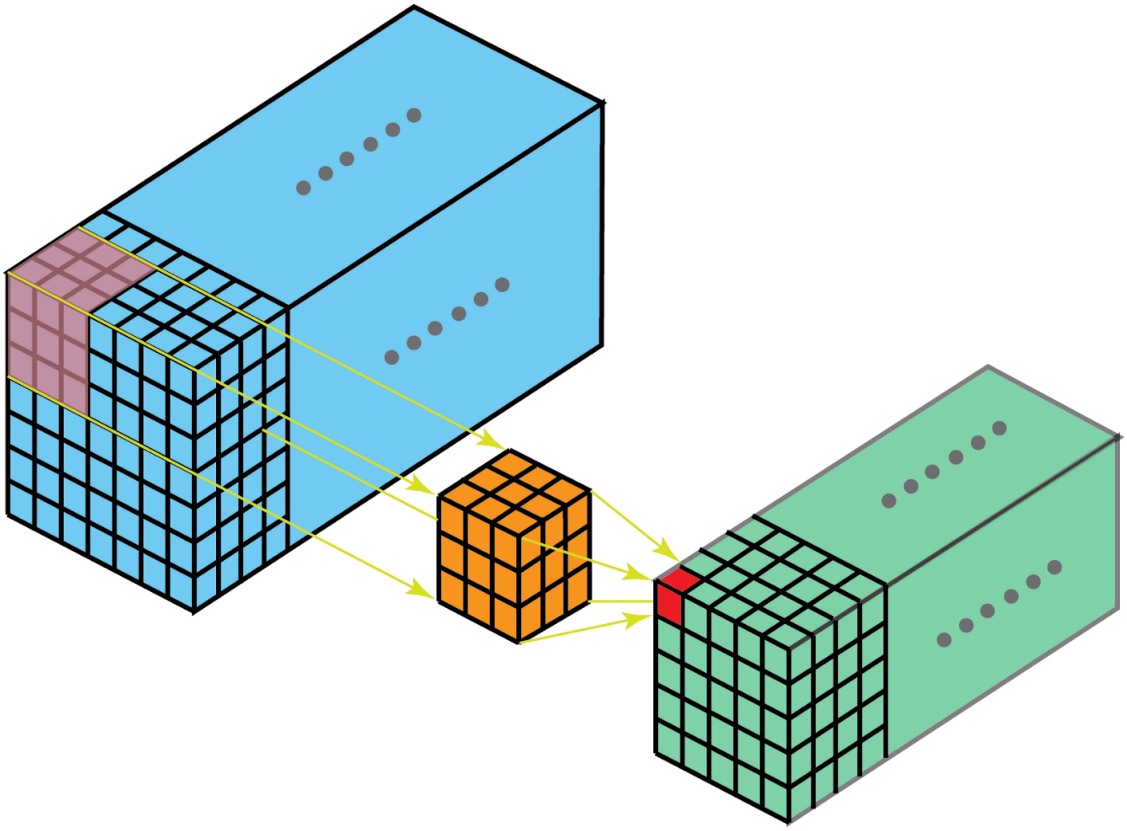
Пример 2d пуллинга:



В случае пуллинга латентных пространств мы будем использовать 3d пуллинги вместа 2d пуллингов, если новая размерность нам будет не подходить из-за все еще большого количества исходов. Соответственно в случае 3d пуллинга могут возникнуть проблемы, что пуллинг

будет проходить межлу различными каналами, кторые могут быть не связаны между собой. В таком случае мы будем использовать 2d пуллинги с большим окном.

Пример 3d пуллинга:

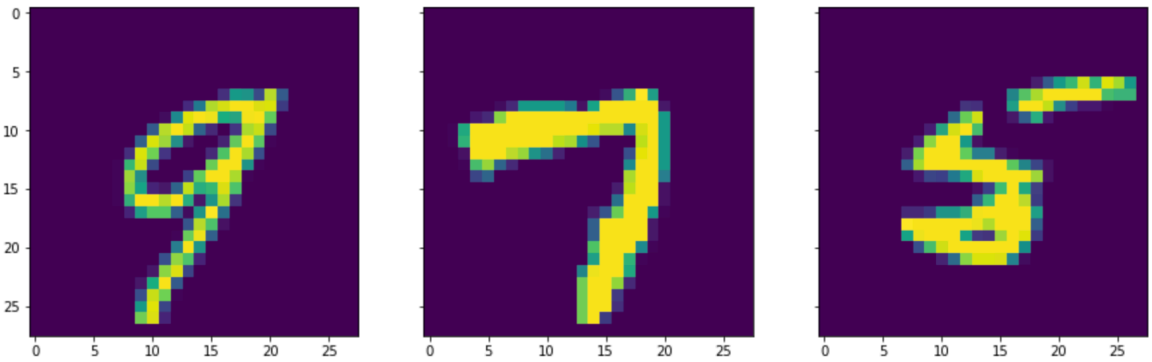


5. Трансформация пространства

Рассмотрим подробнее каждое из пространств и подумаем как именно можно их сужать и какие гиперпараметры могут быть. Здесь мы рассотрим это с логической точки зрения, качетсвенно на практике мы рассмотрим в следующем параграфе

5.1. Входное пространство

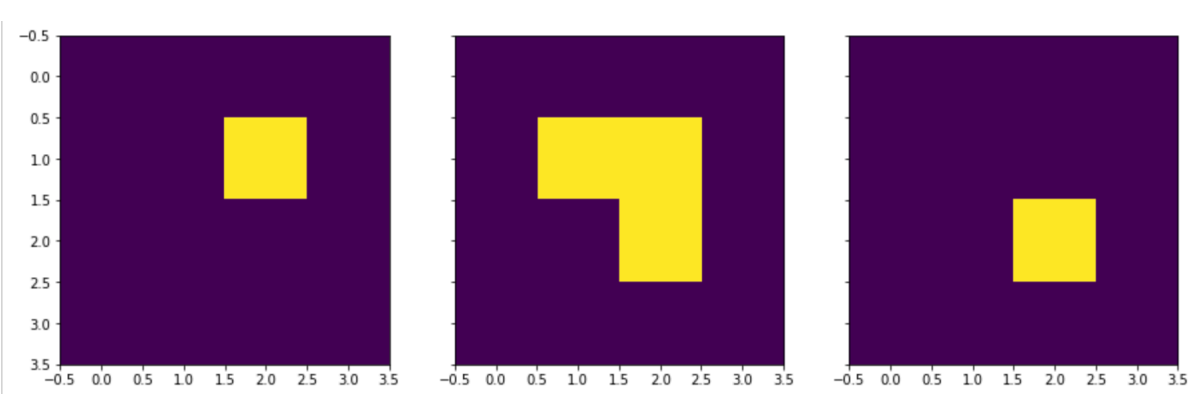
Рассмотрим более подробно сужение на входном пространсве, насколько мы знаем X представляет собой тензор размера (28, 28). При использовании дискретизации на 2 значения (0, 1) полученное пространство имеет размер 2^{28*28}, что очень много, поэтому перед дискретизацией мы будем использовать пуллинги. Рассмотрим примеры входных пространств:



Как видно картинки представляют собой узкие линии значений близких к 1. Соответственно градиент по картинке высокий. Соответственно при использовании усреднения картинка получится размытая и не будет обладать большим количеством информации о том, что было изначально нарисовано на картинке.

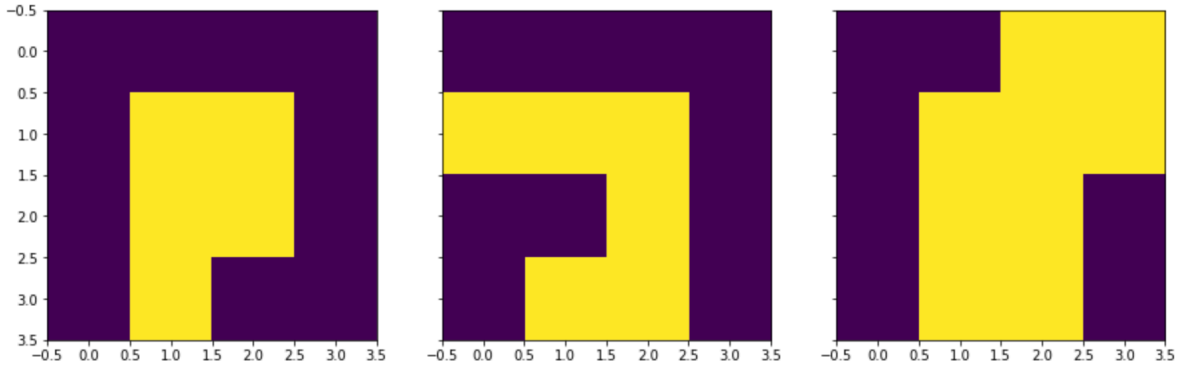
Пример использования $AveragePooling$ с окном (7, 7) с навешенной дискретизацией $discrete$:

$$discrete(x) = \begin{cases} 0, & \text{if } x \leqslant 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{16}$$



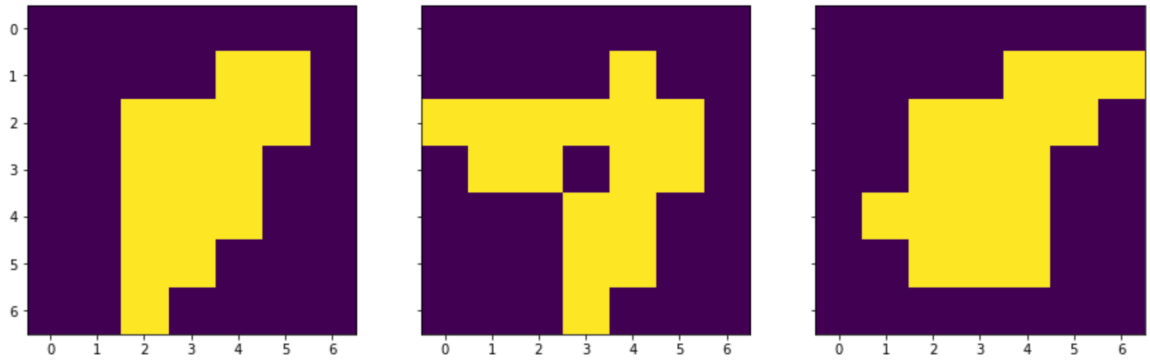
Праклика показывает что полученное представление не переносит много информации с входного пространства. Если выбирать в окне максимум полученное изображение не будет размываться, а значит будет сохранятся больше информации.

Пример использования $MaxPooling$ с окном (7, 7) с навешенной дискретизацией $discrete$:



В качетстве увеличения информации имеет смысл уменьшить окно пуллинга. При этом пространство будет больше и больше вероятность инъективности отображения, но количество перетекаемой информации также увеличится.

Пример использования $MaxPooling$ с окном (4, 4) с навешенной дискретизацией $discrete$:



5.2. Латентное финальное пространство

Здесь мы рассотрим пространство Z_3 которое является выходным слоем сети. Во время обучения от данного слоя берется функция $Softmax$ и уже сравнивается с $onehot$ векторов метки изобажения. Соответственно в пространстве Z_3 нам не важны сами значения, нам важен их порядок и индекс наибольшего значения, так как после применения операции $Softmax$ наибольшее значение из всех станет близки к 1, а все остальные близки к 0

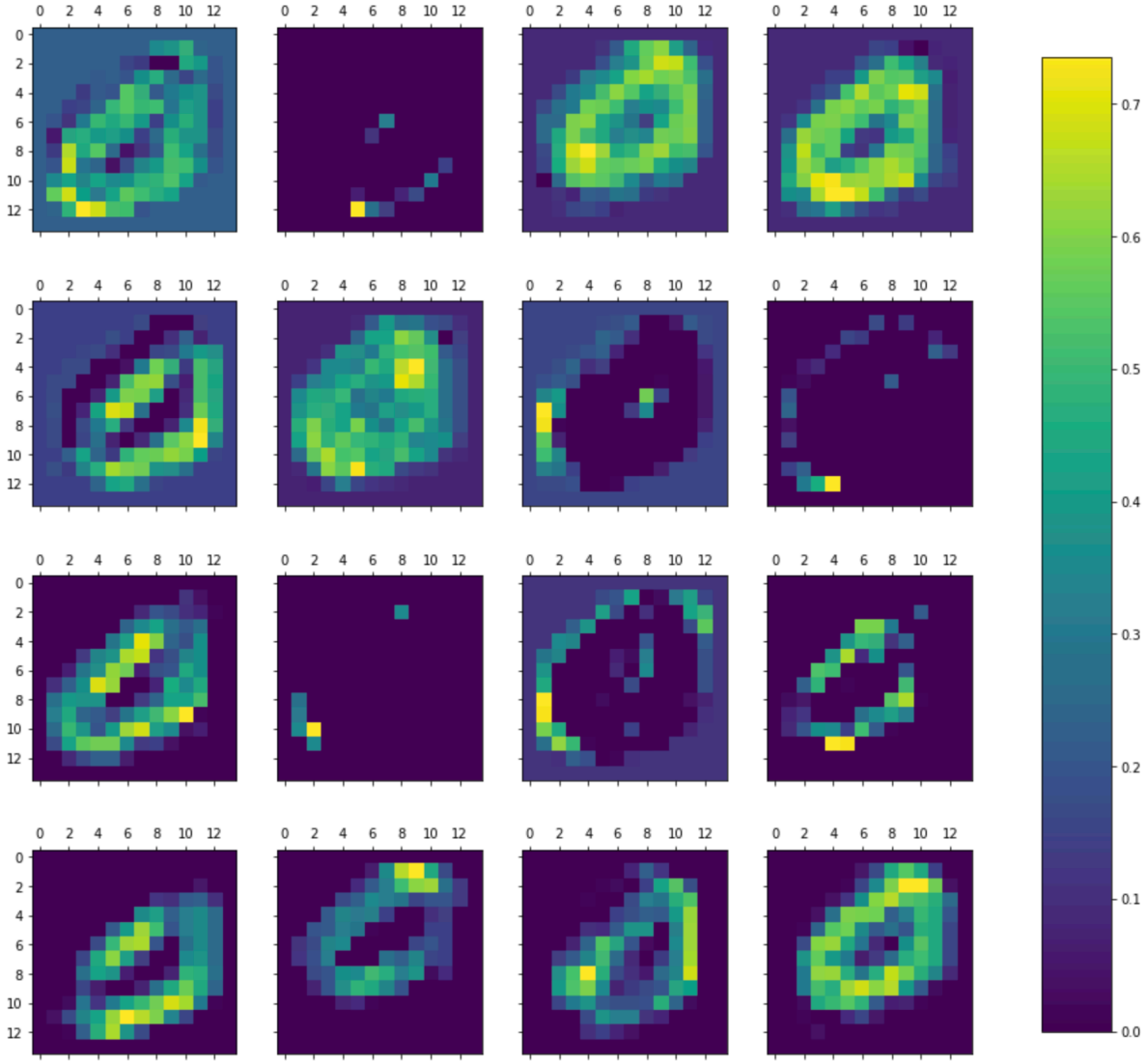
$$softmax(x_i) \approx \begin{cases} 1, & \text{if } i = max\_index \\ 0, & \text{if } i \neq max\_index \end{cases} \tag{17}$$

Поэтому нам не так важно распеделение на финальном слое, как важен порядок. Поэтому преобразование на финальном латентном пространстве будет функция $Softmax$.

5.3. Промежуточные латентные пространства

В качетсве примера возьмем устройство пространства Z_1, пространство Z_2 устроено аналогично, так как оно получается таким же образом, а имеено с помощью сверточного слоя. Напомним что пространство Z_1 имеет размерность $(dataset\_size, 16, 14, 14)$

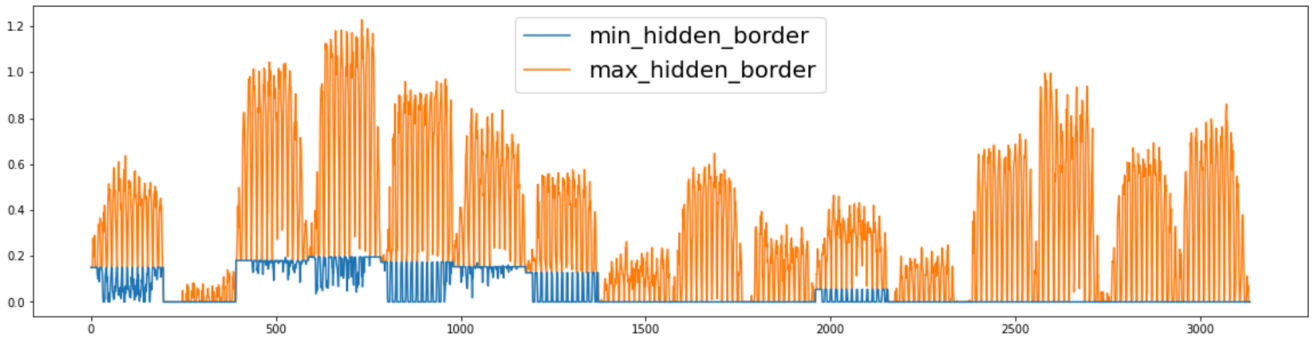
В качестве примера рассмотрим выход $z_1 \in Z_1$ - а именно результат свертки одной из входных картинок. Разложим тензор по всем 16 каналов:



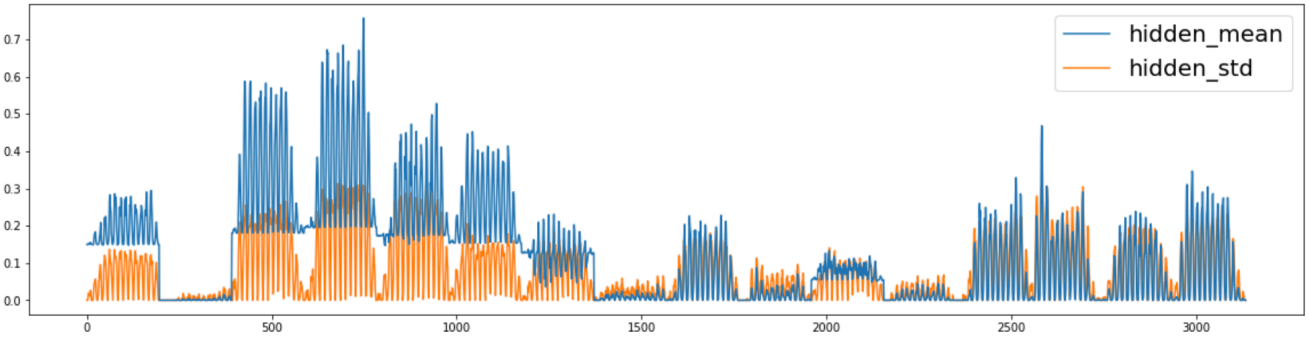
Как видно, каждый из 16ти различных каналов представляет собой такое же изображение, где каждая ячейка показывает как реагирует соответствующее окно на определенный фильтр, который сответствует данному каналу. С точки зрения информации нам важно какие места реагируют на различные свертки-фильтры, то есть нам важно понимать как утроены большие значения по признакам (зеленые, желтые на картинке).

Для дальнейшей дискретизации нам важно как распределены значения в пространстве по каждому призраку. Для этого посчитаем 0.01 квантиль (min_hidden_border) и 0.99 квантиль

(max_hidden_border) для каждого из признаков по выборке:

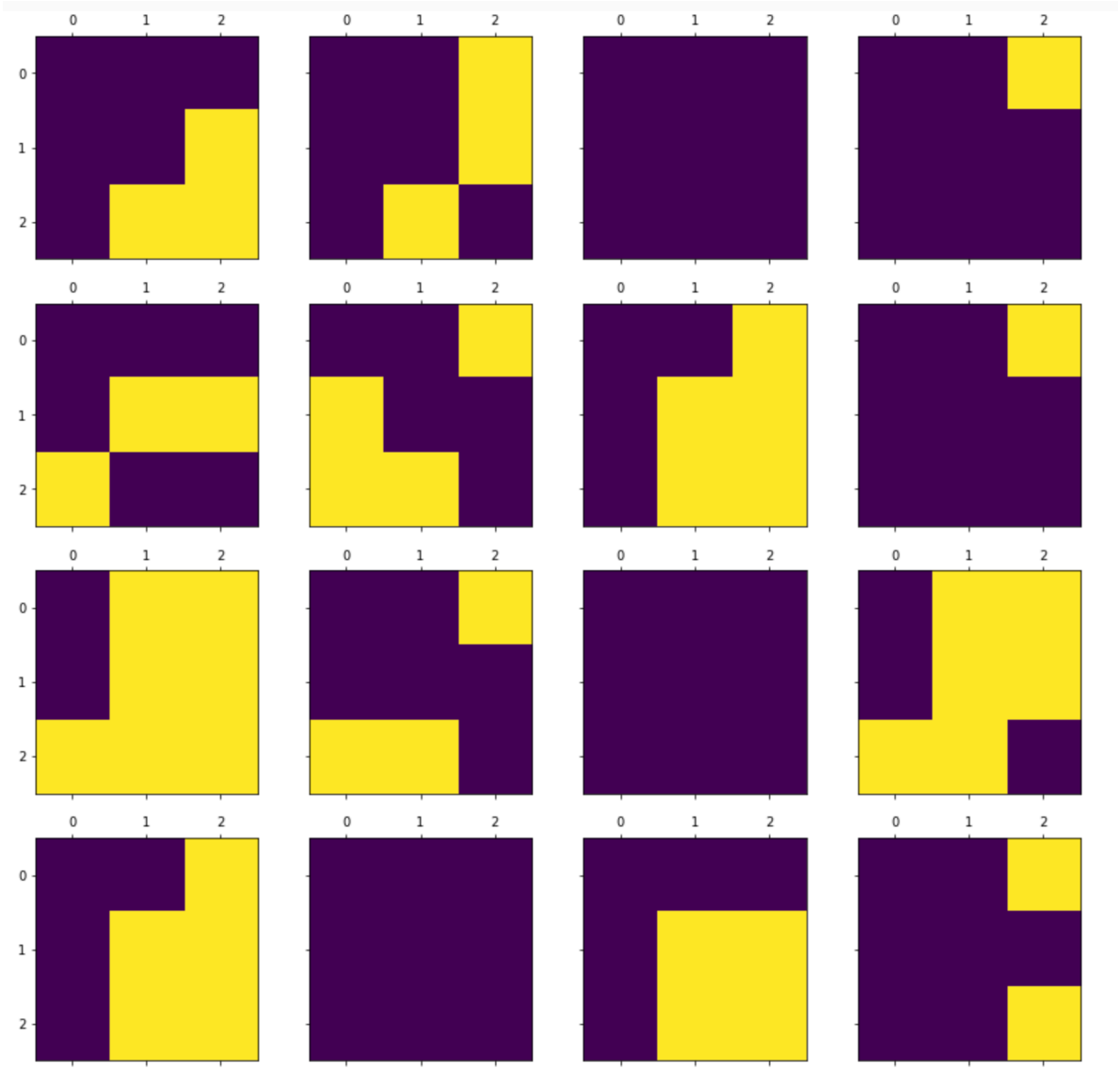


Также посчитаем среднее $hidden\_mean$ и дисперсию $hidden\_std$ по каждому признаку:



Как видно распределение по каждому призаку отличаются, отсюда можно сделать вывод что при дискретизации значения к которым будут приближаться пространства будут для каждого признака различны и расчитываться из соответствующих для них min_hidden_border и max_hidden_border.

Поскольку нам важно понимать как утроены места с большими значениями в качестве пуллинга будет использоваться $maxpool$, поскольку $avgpool$ будет делать значения более размытыми. Пример дискретизации с последовательным применением пуллинга и дискретизации, то есть дискретизация проводится по уже сжатой выборке:



Аналогичную операцию сужения пространства будем проводить с следующем латентным пространством Z_2. Параметры пуллинга остаются гиперпараметрами и устанавливаются на валидации. При этом условие проводить $MaxPool2d$ или $MaxPool3d$ также остается гиперпараметром, так как каналы более независимы друг от друга чем значения внутри каждого канала. Но при этом при смешиваниии каналов дискретизация приводит к меньшему пространству и уменьшается риск инъективности отобрания.

6. Информационные метрики

6.1. Случай без сужения пространства

Для оценки метрики

$$min_Z I[X,Z] - \beta * I[Z,Y] \tag{18}$$

требуется считать совместные информационные эвристики между латентным пространством и входным/выходным, вспомим формулу информации между латентным пространством Z и входным пространством X

$$\begin{gathered} I[X,Z] = -\mathbb{E}_{p(x,z)} \log(\frac{p(x)p(z)}{p(x,z)}) = \\ \sum_{i=0}^{len\_dataset} p_{XZ}(x=x_i, z=z_i) * \log(\frac{p_{XZ}(x=x_i, z=z_i)}{p_X(x=x_i) * p_Z(z=z_i)}) \end{gathered} \tag{19}$$

где

$$\begin{gathered} p_X(x=x_i) = count(discrite(x_i))/len(dataset) \\ p_Z(z=z_i) = count(discrite(z_i))/len(dataset) \\ p_{XZ}(x=x_i, z=z_i) = count(concat(discrite(x_i), discrite(z_i)))/len(dataset) \end{gathered} \tag{20}$$

То есть мы считаем количество элементов эдементов в датасете которые перешли в одни и те же дискретные исходы и на основе этих количеств строим вероятностное пространство. Заметим что свойства вероятностной меры соблюдены: сумма всеъ вероятностей равна 1, мера не отрицательна, соблюдедается аддитивность.

Аналогично считается $I[Y,Z]$:

$$\begin{gathered} I[Y,Z] = -\mathbb{E}_{p(y,z)} \log(\frac{p(y)p(z)}{p(y,z)}) = \\ \sum_{i=0}^{len\_dataset} p_{YZ}(y=y_i, z=z_i) * \log(\frac{p_{YZ}(y=x_i, z=z_i)}{p_Y(y=y_i) * p_Z(z=z_i)}) \end{gathered} \tag{21}$$

только при этом не происходит дискретизация для Y, тк оно и так принимает целые значения от 0 до 9.

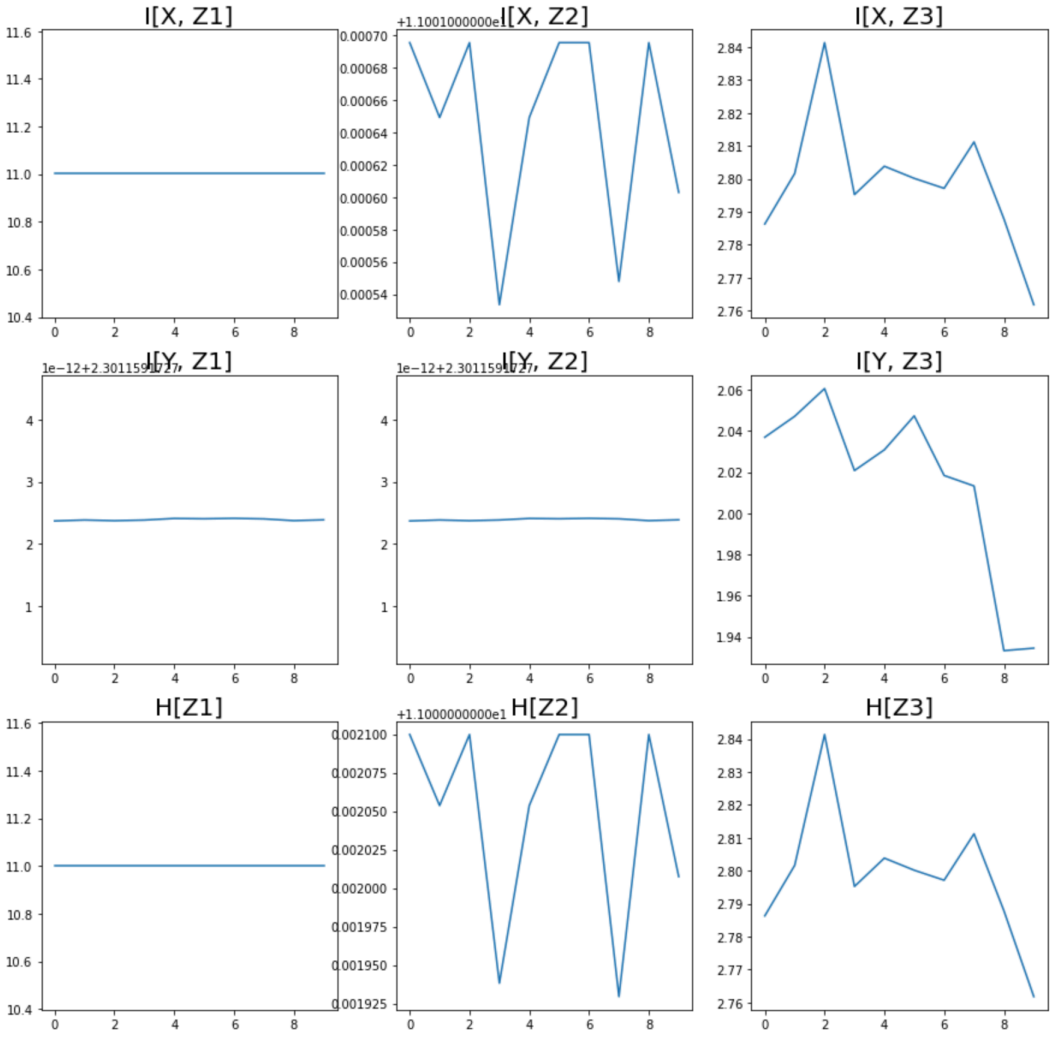
Соответственно если рассматривать пространства p_{XZ}, p_{YZ}, p_X, p_Z без сужения велика вероятность получения равномерного распределения независимо от латентного пространства, то есть

$$\begin{gathered} p_X(x=x_i) = \frac{count(discrite(x_i))}{len(dataset)} \approx \frac{1}{len(dataset)} \\ p_Z(z=z_i) = \frac{count(discrite(z_i))}{len(dataset)} \approx \frac{1}{len(dataset)} \\ p_{XZ}(x=x_i, z=z_i) = \frac{count(concat(discrite(x_i), discrite(z_i)))}{len(dataset)} \approx \frac{1}{len(dataset)} \end{gathered} \tag{22}$$

Это будет для больших пространств пространств, таких как входной слой или первые слои свертки, где также много параметров. Дело в том, что дискретизация в таких случаях будет инъективным отображением как на входном пространстве, так и на латентном, также функция преобразования (часть forward pass) будет инъективной, а значит выполняются такие апроксимации для любых латентных пространств независимо от эпохи обучения. Соответветственно информация будет константной и не будет менятся от эпохи обучения, как и метрика Information Bottleneck (15).

Соответственно для решения этого нам потребуются уменьшение вероятностных пространств, чтобы избавится от инвариантоности и сделать распределение неравномерным. Также без сужение пространства построение вероятностного пространства и подсчет информационных метрик ресурсно затратно, так как требуется хранить большой сет и итерироваться по нему, что долго.

Представим инфмаорционные метрики на графиках в зависимости от эпохи обучения. Всего было 10 эпох обучения, в каждой из итераций строилось новое вероятностное пространство и по нему считались давнные эвристики.



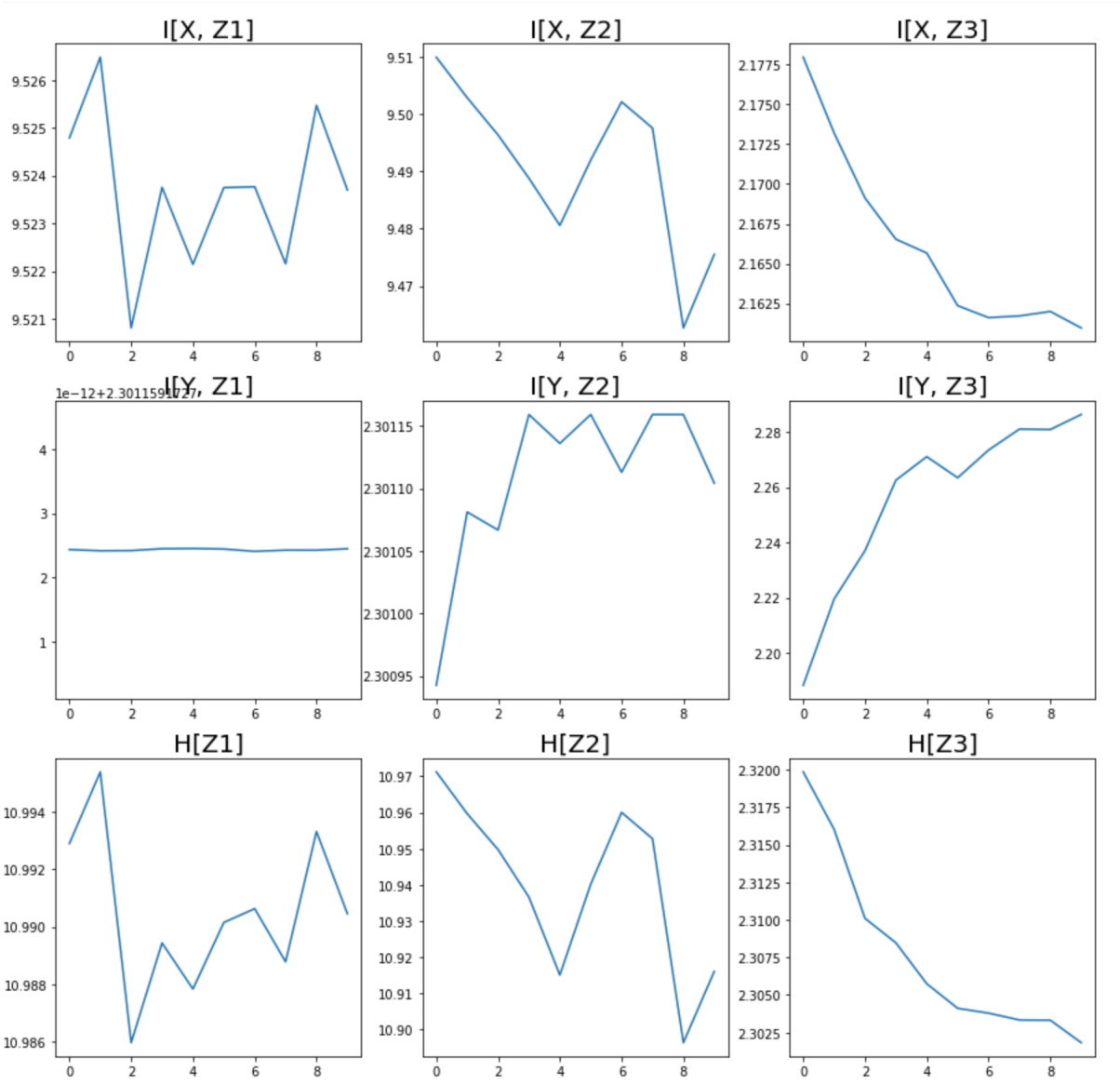
Как можно заметить $H[Z_1], I[X, Z_1], I[Y, Z_1]$ статичны независимо от эпохи. Это происходит из-за того что построенные вероятностные пространства распределены равномерно, это еще раз подтверждает то, что латентное пространство требуется сужать. Также метрики не имею никакую тенденцию. Мы пришли к выводу что задача нейронной сети уменьшение эвристики:

$$I[X, Z] - \beta * I[Z, Y] \tag{23}$$

Здесь же не наблюдается уменьшение метрики $I[X, Z]$ и увеличение метрики $I[Y, Z]$ ни для ни одного латентного пространства Z_1, Z_2, Z_3.

6.2. Случай сужения пространства

Рассмотрим пример с сужением вероятностных пространств и посчитаем на них аналогично информационные эвристики в зависимости от эпохи:



Здесь в данном примере в качестве гиперпараметров возьмем:

1. $waist\_input = MaxPool2d((4,4))(input)$
2. $waist\_z1 = MaxPool3d((1,4,4))(z1)$
3. $waist\_z2 = MaxPool3d((1,3,3))(z2)$
4. $waist\_z3 = Softmax(z3)$

То есть в промежуточных латентных слоях не смешивались каналы между собой.
Здесь проще всего рассмотреть как ведут себя информация между входным пространством и латентным $I[X,Z]$ и энтропия латентных пространств $H[Z]$. Как видно на графике данные эвристики уменьшаются по мере обучения, причем чем проще латентное пространство устроено тем более функция от эпохи монотонная. Например $I[X,Z_3]$ и $H[Z_3]$ имеют более выраженое уменьшение с увеличением эпохи чем аналогичные эвристики только для Z_2 и Z_1. Аналогичное можно сказать про информацию между выходом и латентными пространствами $I[Y,Z]$. Чем больше Z_i тем более монотона устроена функция. Только при этом $I[X,Z$ уменьшается в то время как $I[Y,Z]$ увеличивается. Более вырожденная монотонность происходит по причине что пространсво Z_3 устроено проще чем другие латентные пространства. $Softmax(z_3)$ стремится к $one\_hot(y)$ для любых соответсвующих $z_3 \in Z_3, y \in Y$. Таким образом в хорошо обученной сети:

$$
\begin{aligned}
I[Y,Z] = \sum_{i=0}^{len\_dataset} p_{YZ}(y=y_i, z=z_i) * \log(\frac{p_{YZ}(y=x_i, z=z_i)}{p_Y(y=y_i) * p_Z(z=z_i)}) \approx \\
\sum_{i=0}^{len\_dataset} p_Y(y=y_i) * \log(\frac{p_Y(y=y_i)}{p_Y(y=y_i) * p_Y(y=y_i)} = \\
- \sum_{i=0}^{len\_dataset} p_Y(y=y_i) * \log(p_Y(y=y_i)) = H(Y) \approx \\
\sum_{i=1}^{10} 0.1 * \log(10) = \log(10) \approx 2.3
\end{aligned}
\tag{24}
$$

Как видно в текущем примере $I[Y,Z]$ не достигает максимальной оценки. Увеличение метрики $I[Y,Z]$ и уменьшение метрики $I[X,Z]$ подтвержает нашу гипотезу что метрику Information Bottleneck можно использовать как лосс функции, так как функция $I[X,Z] - \beta * I[Z,Y], \beta > 0$ уменьшается с улучшением сети. При этом информационные метрики для Z_1 устроены самым непонятным образом. Это связано с тем что данное латентное пространство наиболее сложное, и в нем наибольшее количество параметров:

1. $waist\_input.shape = (1, 7, 7) \Rightarrow len(input\_params) = 2^{1*7*7} = 2^{49}$
2. $waist\_z1.shape = (16, 3, 3) \Rightarrow len(z1\_params) = 2^{16*3*3} = 2^{144}$
3. $waist\_z2.shape = (32, 2, 2) \Rightarrow len(z2\_params) = 2^{32*2*2} = 2^{128}$
4. $waist\_z3.shape = (10) \Rightarrow len(z3\_params) = 2^{10}$

6.3. Валидация гиперпарметров

Проведем несколько экспериментов, сравнив информационные метрики для различных вероятностных пространств. Для этого первым экспериментом смешаем каналы для пространств Z_2 и Z_3, то есть MaxPooling будет проводиться также и по каналам. Этот эксперимент имеет смысл, так как в предъидушем примере латентные пространства имеют слишком большую размерность и как вывод сложные информационные представления. Соответственно смешиваение по каналам позволит уменьшить размерность пространства. В еще одном примере уменьшение размерности пространства будем осуществлять засчет увеличение окна 2d свертк, это позволит не смешивать каналы между собой. Для этого рассмотрим 3 примера с другими сверками.

1 пример:

1. $waist\_input = MaxPool2d((4, 4))(input)$
2. $waist\_z1 = MaxPool3d((2, 4, 4))(z1)$
3. $waist\_z2 = MaxPool3d((2, 3, 3))(z2)$
4. $waist\_z3 = Softmax(z3)$

2 пример:

1. $waist\_input = MaxPool2d((4, 4))(input)$
2. $waist\_z1 = MaxPool3d((4, 4, 4))(z1)$
3. $waist\_z2 = MaxPool3d((4, 3, 3))(z2)$
4. $waist\_z3 = Softmax(z3)$

3 пример:

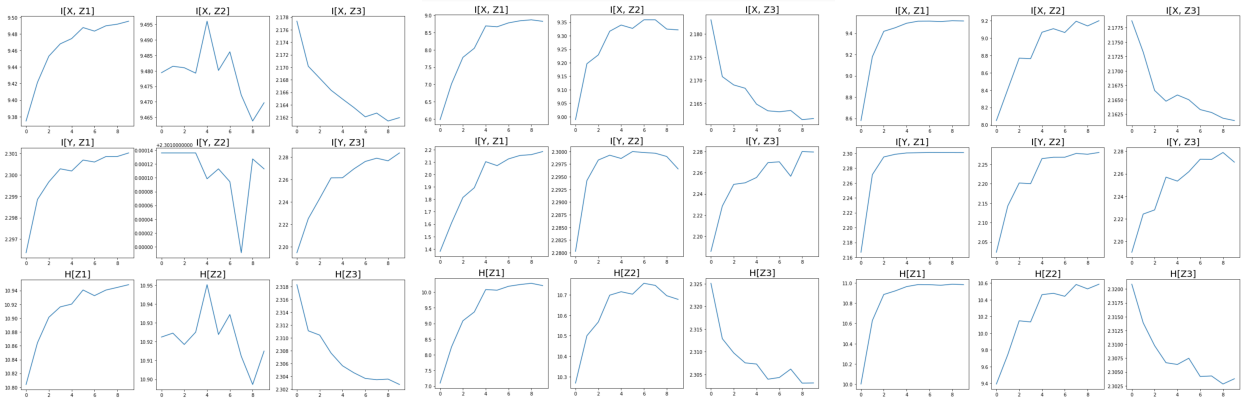
1. $waist\_input = MaxPool2d((4, 4))(input)$

2. $waist\_z1 = MaxPool3d((1,7,7))(z1)$

3. $waist\_z2 = MaxPool3d((1,7,7))(z2)$

4. $waist\_z3 = Softmax(z3)$

и посчитаем для новых вероятностных пространств информационные метрики. Примеры 1-3 расположены слева направа на графиках. Первое на что стоит обратить внивание - это воз-



растание метрики $I[X,Z]$ для всех примеров. Аналогично возрастает $I[Z]$. Это происходит по той причине, что значения на латентных слоях в обученных сетках распределены более равномерно. Это происходит по причине, что в необученной сети значения с большим значениями имеют большую вероятность в отличие от значений с маленькими значениями. Это происходит потому что беря максимум по окну с равномерно распределенными значениями максимум будет с большей вероятностью иметь большие значения

$$X_{max} = max(X_1, X_2, ...X_n) \tag{25}$$

где X_i независимо одинаково распределеные случайные велечины. $X_i \in Uniform(0,1)$, тогда

$$p(X_{max} < \alpha) = p(X_1 < \alpha) * ... * p(X_n < \alpha) = \alpha^n \to 0 \tag{26}$$

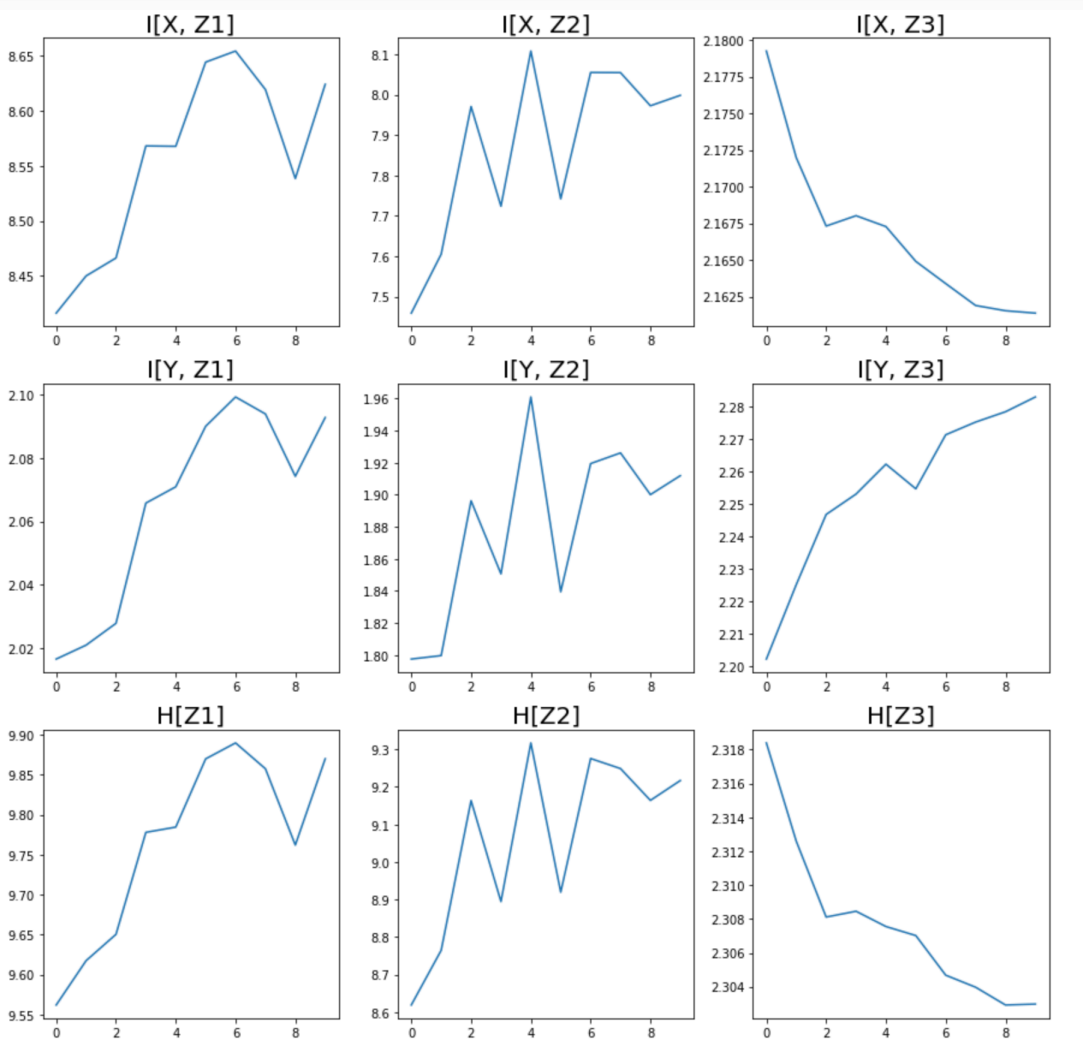
Соответветственно брав максимум по большому окну со смешанными каналами на плохо обученной сети расределение будет неравномерным. При обучении сети исходные латентные пространства имеют более выраженные предсавления: разрешенная матрица, где ненулевые значения обозначают реакцию на конкретные свертки - соответственно брав максимум по большому окну между несколькими каналами на выходе получится наибольшая реакция на свертку, что имеет равномерное распределение, поэтому с увеличением эпохи энтропия увеличивается и как следствие $H[Z]$ и $I[X,Z]$. Соответственно для данных гиперпараметров метика IB не будет уменьшаться с увеличением эпохи.

Для разрешение проблемы неравномерного распределения воспользуемся не maxpooling a avgpooling, то есть будем усреднять а не максимизировать значения в окне. При усреднении неравенство (26) уже не будет действовать, более того среднее у размеренной матрицы стремится к 0, соответственно ситуация с энтропией должны быть симетрична как в прошлом случае. Построим новый пример с нарда новыми гиперпараметрами.

4 пример:

1. $waist\_input = MaxPool2d((4,4))(input)$
2. $waist\_z1 = ArgPool3d((1,7,7))(z1)$
3. $waist\_z2 = ArgPool3d((1,7,7))(z2)$
4. $waist\_z3 = Softmax(z3)$

Расмтрим для данного эксперимента полученную информационную маску



Заметим, что возрастание функций $I[X,Z]$ и $I[Z]$ стало менее мнотонно, но при этом общий характер возрастания при увеличении эпохи остался, что требует большего изучения.

7. Заключение

Сделаем выводы о проведенных экспериментах. Изначально мы пришли к выводу что задача обучения нейронной сети это решение оптимизационной задачи

$$min_Z I[X, Z] - \beta * I[Z, Y] \tag{27}$$

То есть мы хотим минимизировать взаимную информацию между входным пространством и латентным, при этом максимизовать общую информацию между выходом и латентым пространством. Изначально мы хотели проверить насколько эта гипотеза верна на практике. Тезисно приведем выводы которые были сделаны на протяжение работы:

1. Рассмотрев слои, был сделан вывод, что значения по отдельным признакам зависят друг от друга, также значения по каждому признаку распределены нестандартно, поэтому сузим пространства до дискретных и вероятностнь будем оценивать количественно.

2. Сужение должно быть оптимальным, а именно быть не инъективнм, чтобы не быть равномернораспределеным и изменятся от эпохи к эпохе, при этом сохранять информацию о изначальном пространстве.

3. Для сужений принято решение использовать конкатенацию пуллингов и дискретизацию, при этом на выходном слое использовать softmax.

4. Доказано что при сужении не достаточно использовать только дискретизацию, так как размерность полученных пространств слишком большая из-за чего отображение получается инъективно и операция построения вероятностоного пространства слишком долгая.

5. В качестве операции понижения размерности пуллинги сохраняют информацию, так как операция пуллинг часто используется как слои в нейронных сетях для понижения размерности, соответственно через них проходит нужжная информация.

6. Всего в сети 3 латентных пространства, которые последовательно упрощаются, чем сложнее пространство тем более непредстказуемо ведут себя информационные метрики на них.

7. Финальное латентное пространство подтверждает выдвинутую в начале гипотезу и для этого пространства метрика $I[X, Z]$ уменьшается, а $I[Y, Z]$ увеличивается с эпо-

хой обучения, а значит общая метрика (27) (Information Bottleneck) также будет уменьшаться с эпохой.

8. При этом информационнык метрики на первых латентных пространствах ведут себя непредсказуемо. Если уменьшать пространство немного, то метрики либо меняются очень слабо либо хаотично, что говорит о инъективности сужения. При этом сжав слишком сильно пространство все информационные метрики: $I[X, Z]$, $I[Y, Z]$, $H[Z]$ увеличиваются с эпохой обучения.

9. Одна из причин поведения в предъиущем пункте - несохранения распределения значений при пуллинге, то есть при обучении латентное пространство становится более равномерно распределенным, отчешл все метрики возрастают, данная проблема требует дополнительного исследования.

В данной работе были подробно рассмотрены поведения информационных метрик в динамике обучения модели, в частности была подробна рассмотрена небольшая свертночная нейронная сеть, подробно рассмотрено поведение различных слоев данной сети и статистически оценено с вероятностной точки зрения. Также в данной работе построен универсальный способ подсчета информации отдельных слоев так и совместной информации для любого состояния любой нейронной сети. Была четко поставлена гипотеза, построенная на теоретических знаниях из релевантной литературы и проверена в действии на практике.
Все практические вычисления описаны в открытом репозитории
https://github.com/nikolaev1ma/bottleneck_information

Список литературы

[1] Andreas Kirsch, Clare Lyle, Yarin Gal (2021) *Unpacking Information Bottlenecks: Surrogate Objectives for Deep Learning*, Department of Computer Science University of Oxford

[2] Naftali Tishby, Noga Zaslavsky (2015) *Deep Learning and the Information Bottleneck Principle*