онлайн-курс

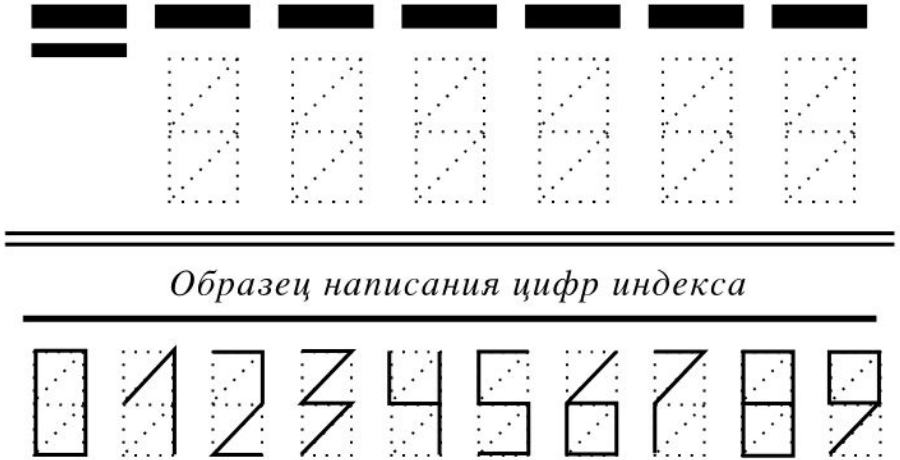
СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Очень большие СНС

LeNet (1998)

Проблема из жизни (не нашей)

В США индекс на конвертах пишется от руки. Не то, что у нас:



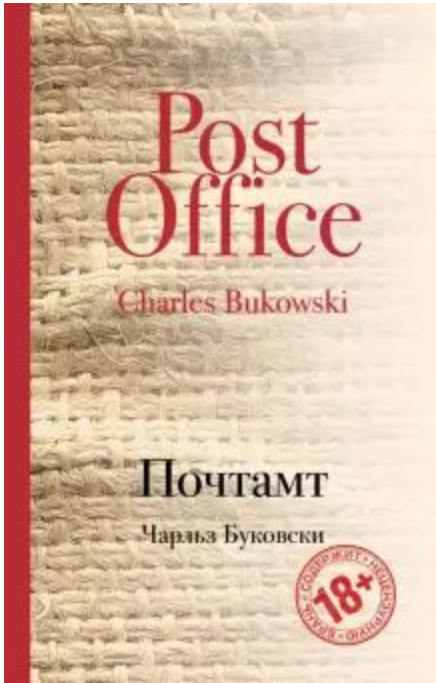
Проблема из жизни (не нашей)

Возникает проблема: **как быстро распознавать индекс?**

Вручную?

Это нереально. Об этом написано (18+) в этой книге:

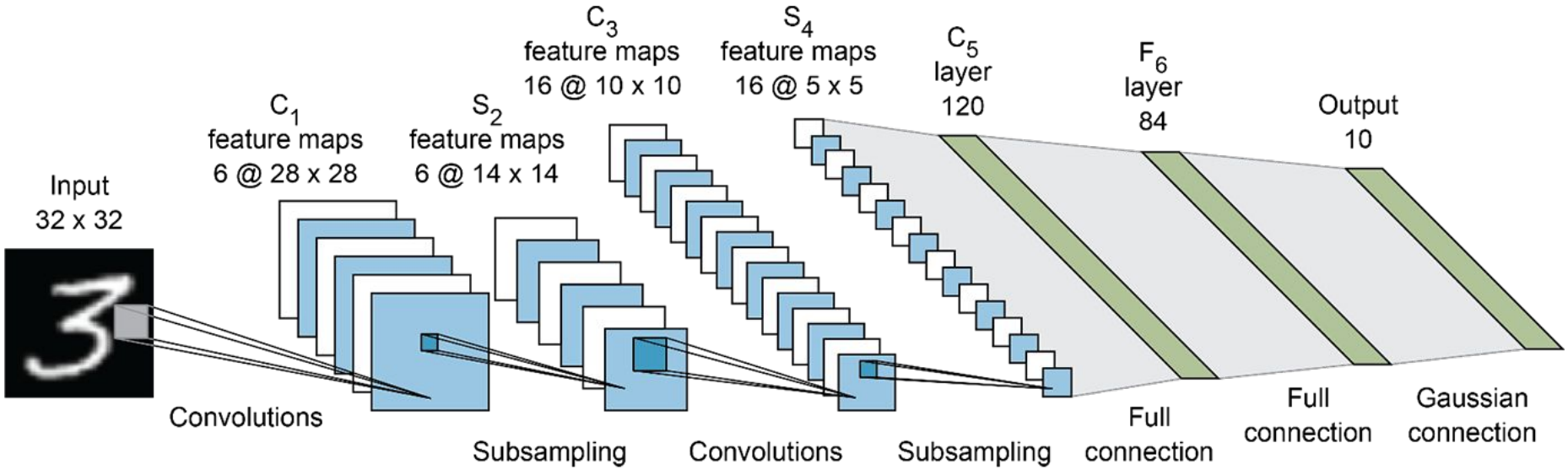
Собственно, все матерные слова в этой книге принадлежат главе про сортировку писем. Ночью. В воскресенье. После употребления алкоголя.



Архитектура и назначение сети

LeNet изначально проектировалась для распознавания рукописных цифр (поэтому у неё 10 выходов).

Подсчитаем общее количество весов сети LeNet.



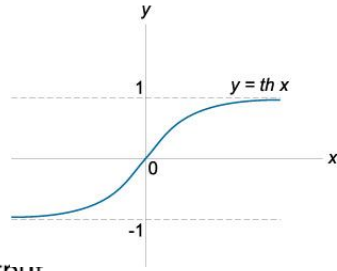
Количество весов сети LeNet

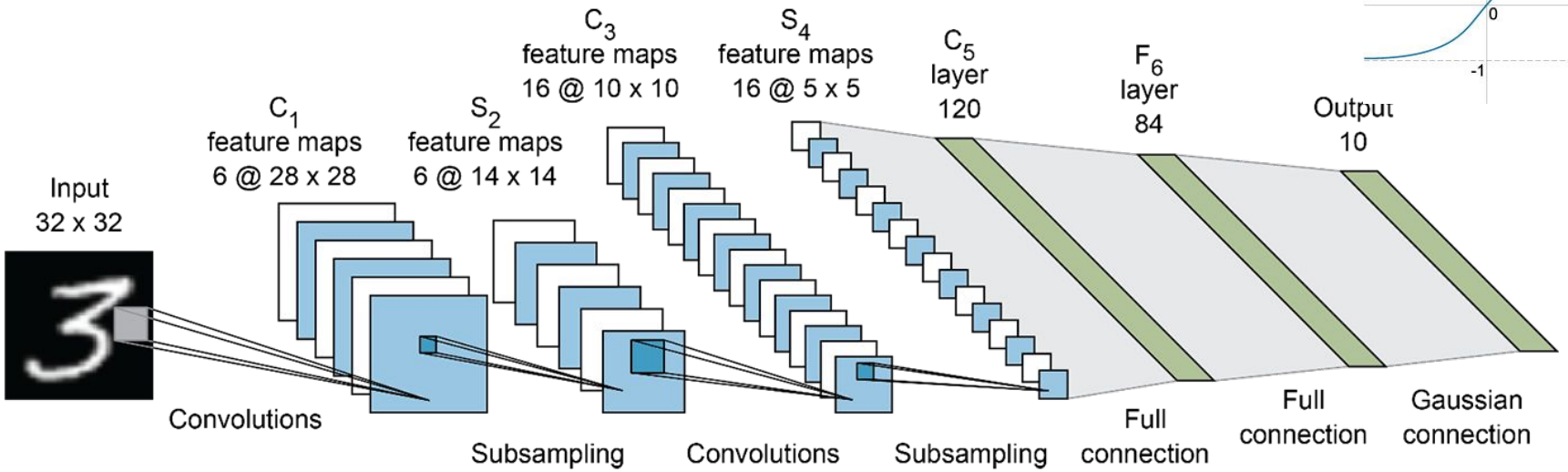
Слой C_1: использовалось 6 фильтров, размера 5x5 (так как 32-5+1=28).

Тут весов 6*5*5=150.

После слоя C_1 к результатам свёртки применяется функция активации tanh(x).

Слой S_2: слой пулинга, тут нет весов.





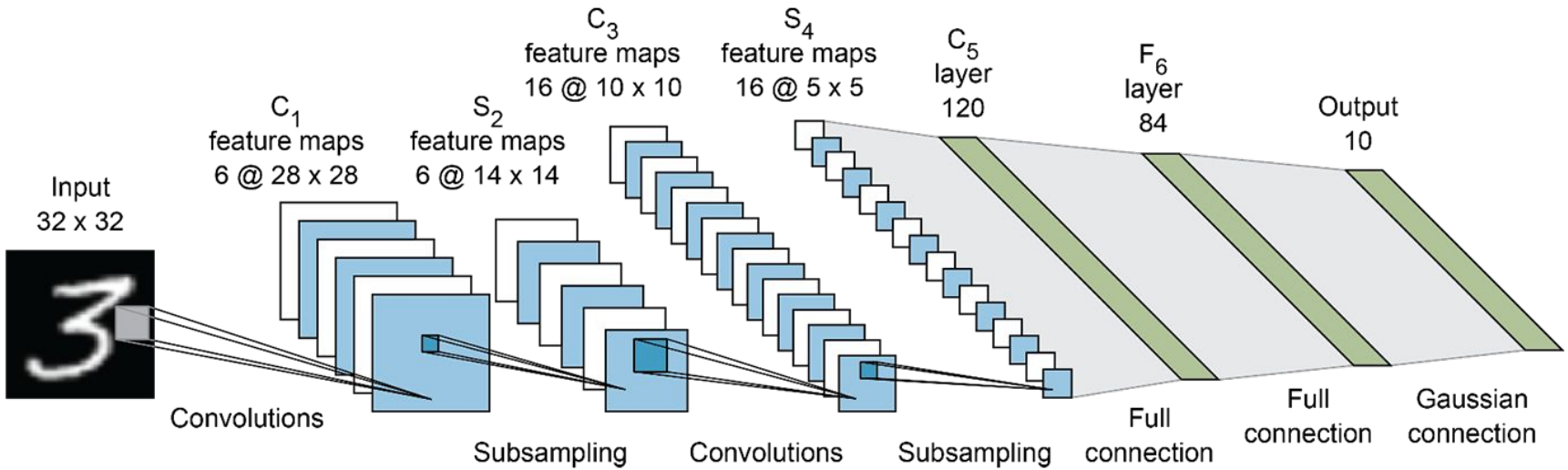
Количество весов сети LeNet

Слой C_3: использовалось 16 фильтров, размера 5x5 (так как 14-5+1=10).

И каждый фильтр имеет 6 каналов. Тут весов 16*6*5*5=2400.

После слоя C_3 к результатам свертки применяется функция активации tanh(x).

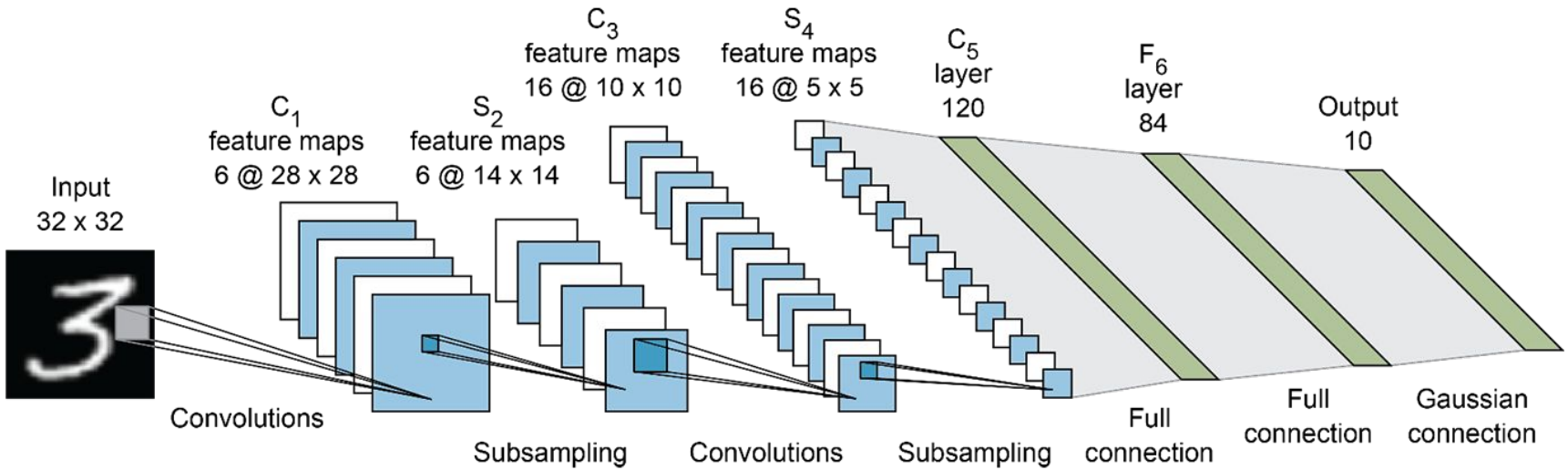
Слой S_4: слой пулинга, тут нет весов.



Количество весов сети LeNet

Слой C_5: его 120 нейронов полносвязно соединены со всеми 16*5*5=400 нейронами последнего сверточного слоя. Следовательно, будет 120*400=48000 весов-связей, плюс ещё 120 весов-смещений.

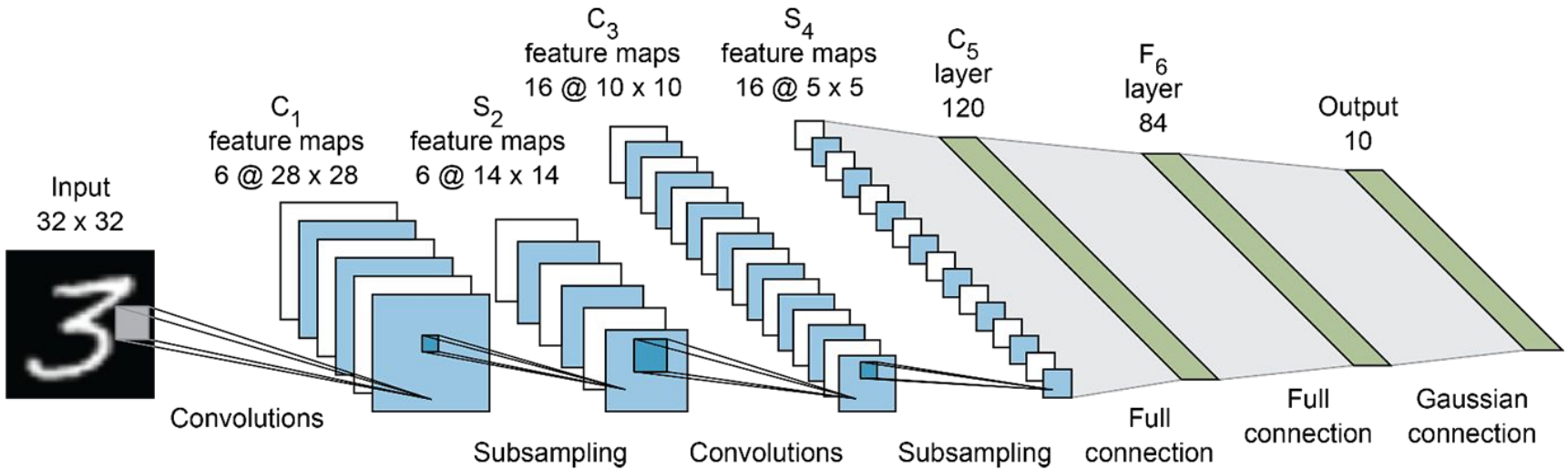
Для выходов из слоя C_5 применяется функция активации tanh(x).



Количество весов сети LeNet

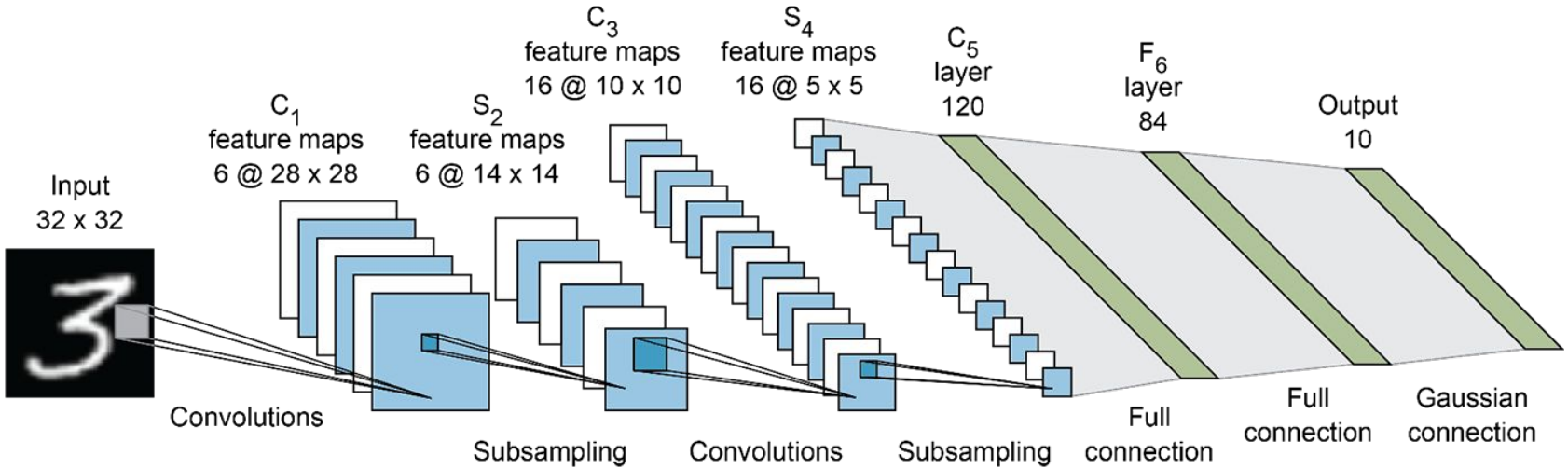
Слой F_6: его 84 нейрона полносвязно соединены со всеми 120 нейронами предыдущего слоя. Следовательно, будет 84*120=10080 весов-связей между слоями. Плюс ещё 84 весов-смещений.

Для выходов из слоя F_5 применяется функция активации tanh(x).



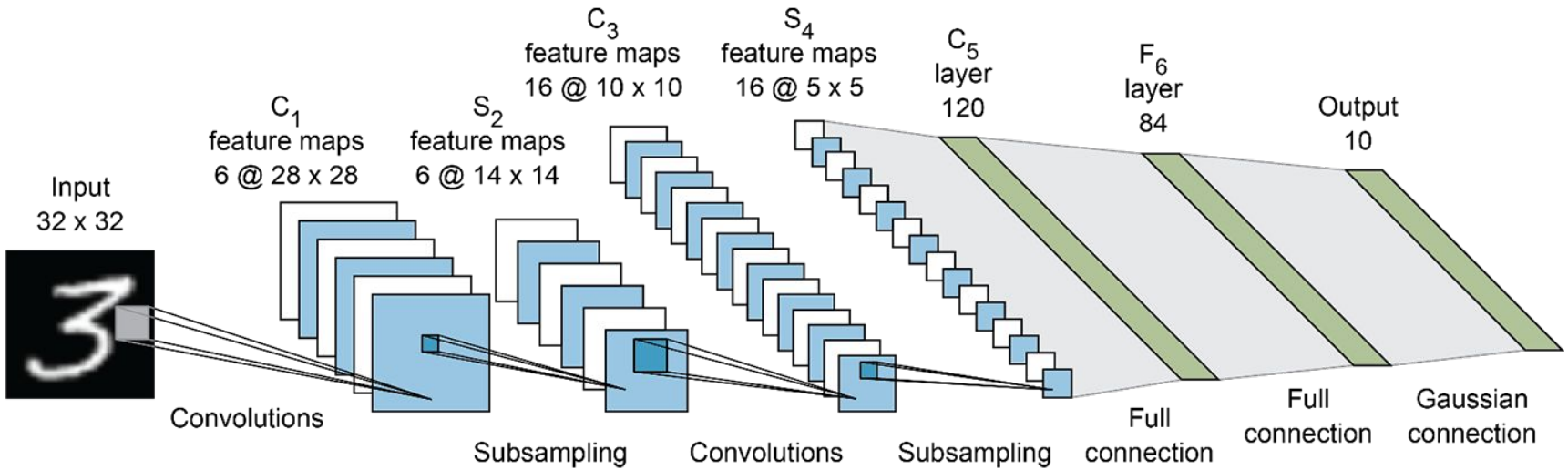
Количество весов сети LeNet

Выходной слой: его 10 нейронов полносвязно соединены со всеми 84 нейронами предыдущего сверточного слоя. Следовательно, будет 10*84=840 весов-связей и 10 весов-смещений. Для выходов из слоя применяется функция активации tanh(x). Для получения вероятностей принадлежности к классам 0...9 применяется преобразование softmax.



Количество весов сети LeNet

В итоге получаем, что сеть LeNet содержит 150+2400=2550 весов в свёрточных слоях, а также 48000+1080+840=49920 весов-связей и 120+84+10=214 весов-смещений в полносвязных слоях.
Всего **52684** веса.



VGG (2014)

Эта сеть была придумана для...

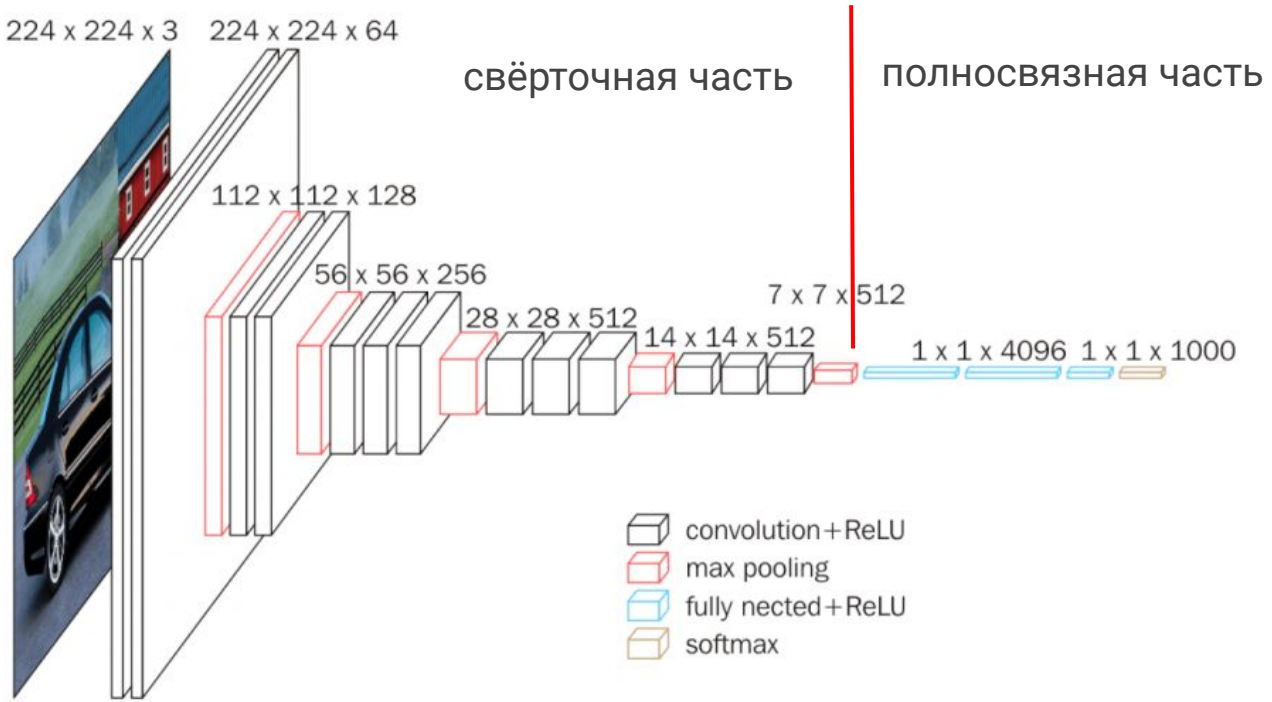
… классификации картинок из датасета ImageNet (миллионы картинок, тысячи классов). И показала очень низкую (по тем временам) вероятность ошибки предсказания (7%).

Она очень толстая, для её тренировки просто необходимо применять все нетривиальные техники: дропаут, нормализация по мини-батчам и т.д.

Здесь показаны размеры свёрточных слоёв.

Всюду используются фильтры 3x3.

Значит, в каждом сверточном слое используется **padding=1**!

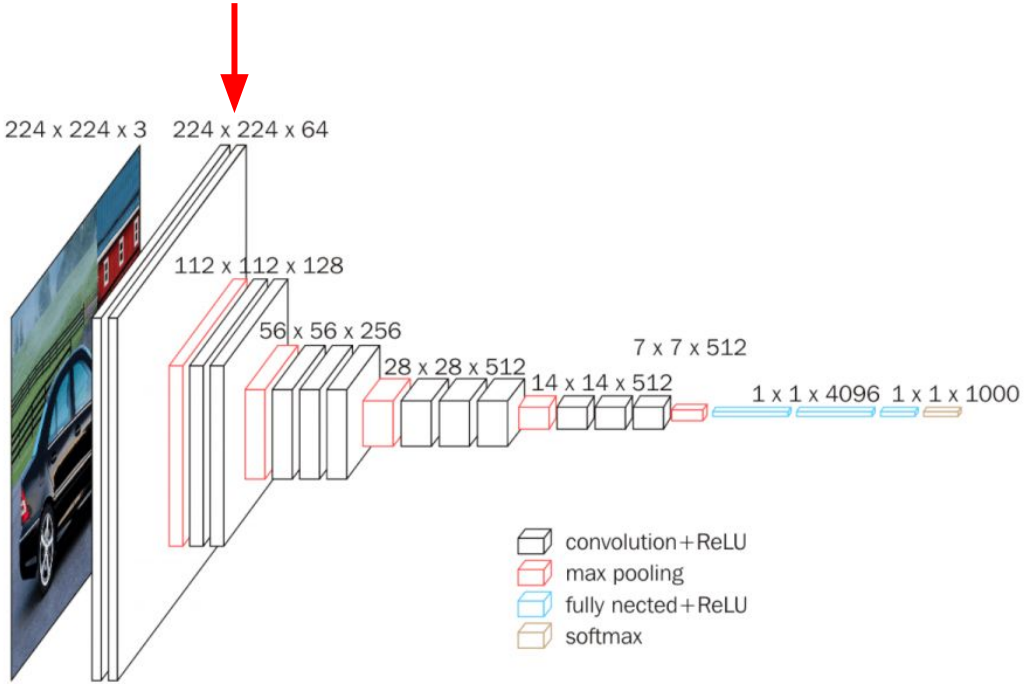


Посчитаем количество весов в слоях VGG

Будем пользоваться этой картинкой.

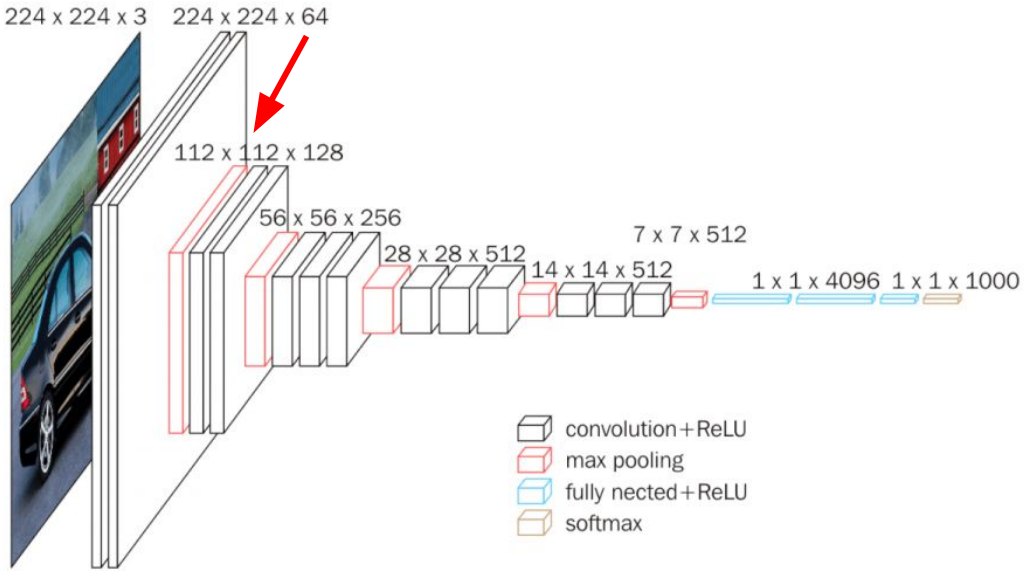
1. Применяем 64 фильтра размера 3x3, и каждый фильтр 3-канальный.

Следовательно, тут 64*3*3*3=**1728** весов для тренировки.



Посчитаем количество весов в слоях VGG

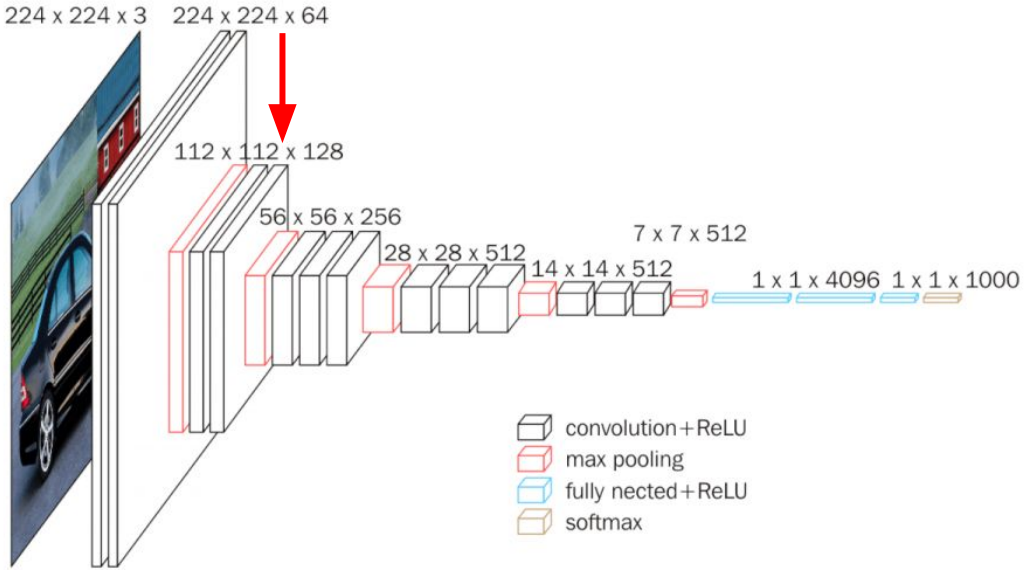
2. Здесь применяется 2-пулинг, размерность изображений уменьшается до 112x112.



Посчитаем количество весов в слоях VGG

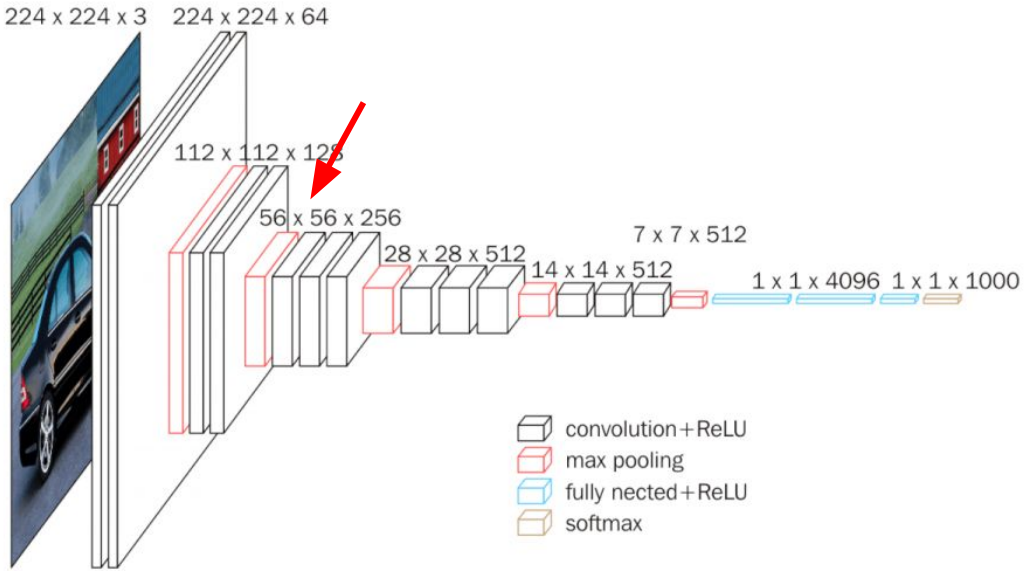
3. Применяем 128 фильтров размера 3x3, и каждый фильтр 64-канальный.

Следовательно, тут 128*3*3*64=**73728** весов для тренировки.



Посчитаем количество весов в слоях VGG

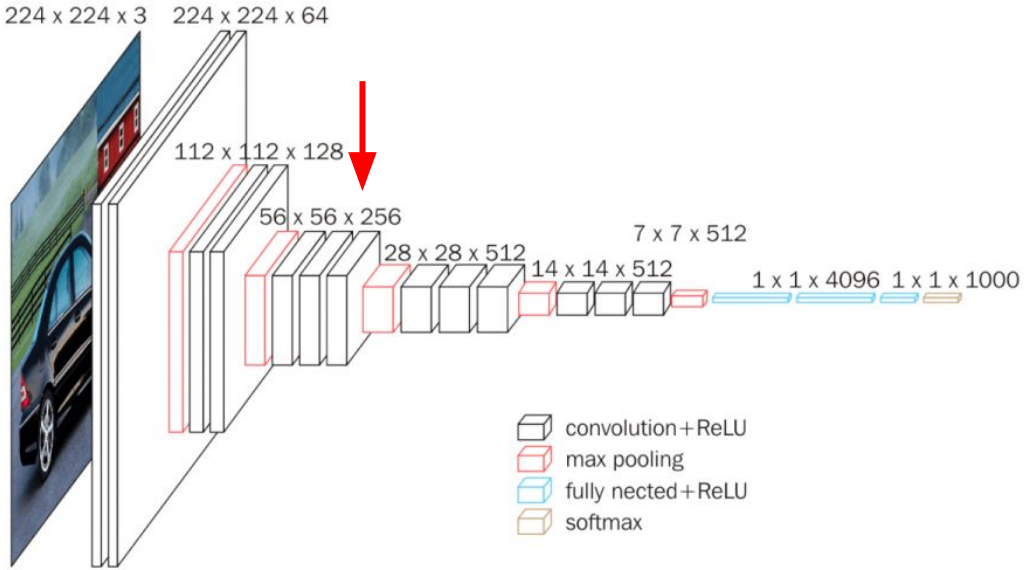
4. Здесь применяется 2-пулинг, размерность изображений уменьшается до 56x56.



Посчитаем количество весов в слоях VGG

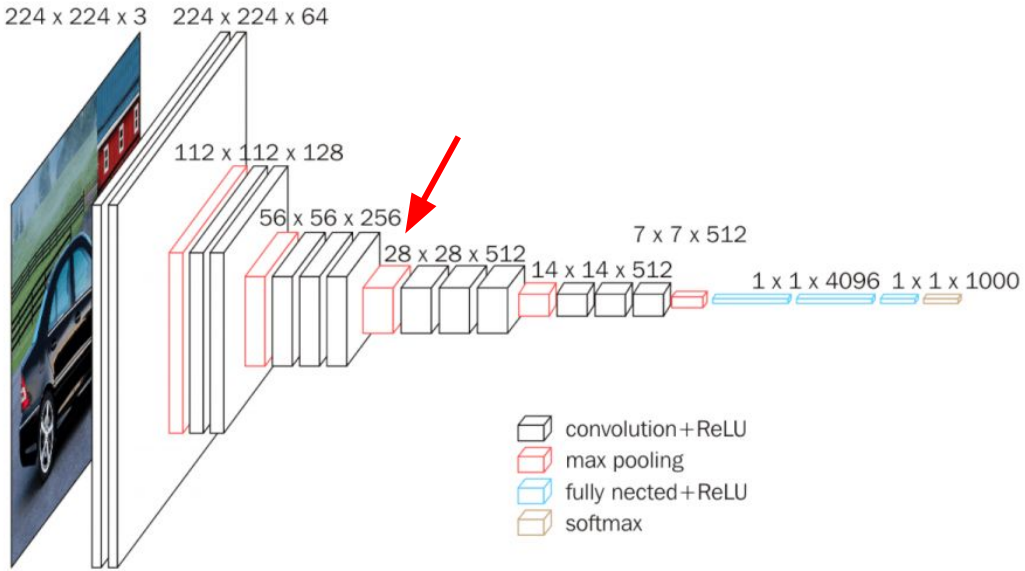
5. Применяем 256 фильтров размера 3x3,
и каждый фильтр 128-канальный.

Следовательно, тут 256*3*3*128=**294912**
весов для тренировки.



Посчитаем количество весов в слоях VGG

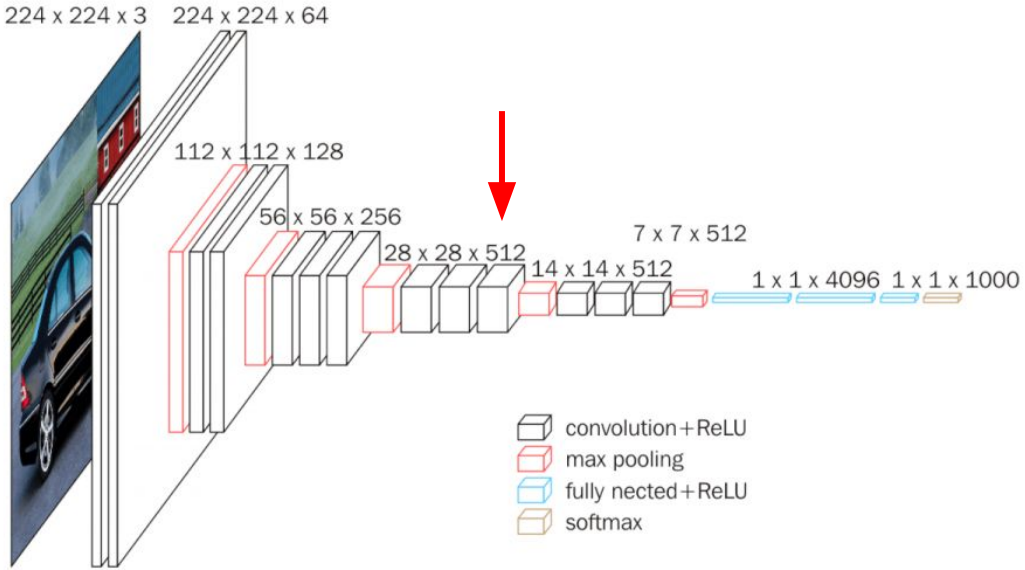
6. Здесь применяется 2-пулинг, размерность изображений уменьшается до 28x28.



Посчитаем количество весов в слоях VGG

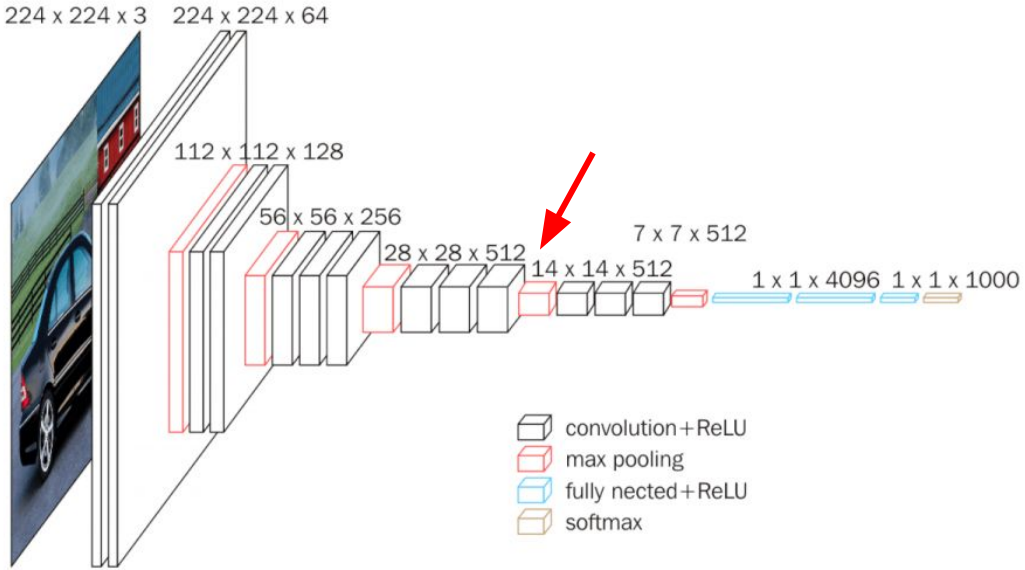
7. Применяем 512 фильтров размера 3x3,
и каждый фильтр 256-канальный.

Следовательно, тут 512*3*3*256=**1179648**
весов для тренировки.



Посчитаем количество весов в слоях VGG

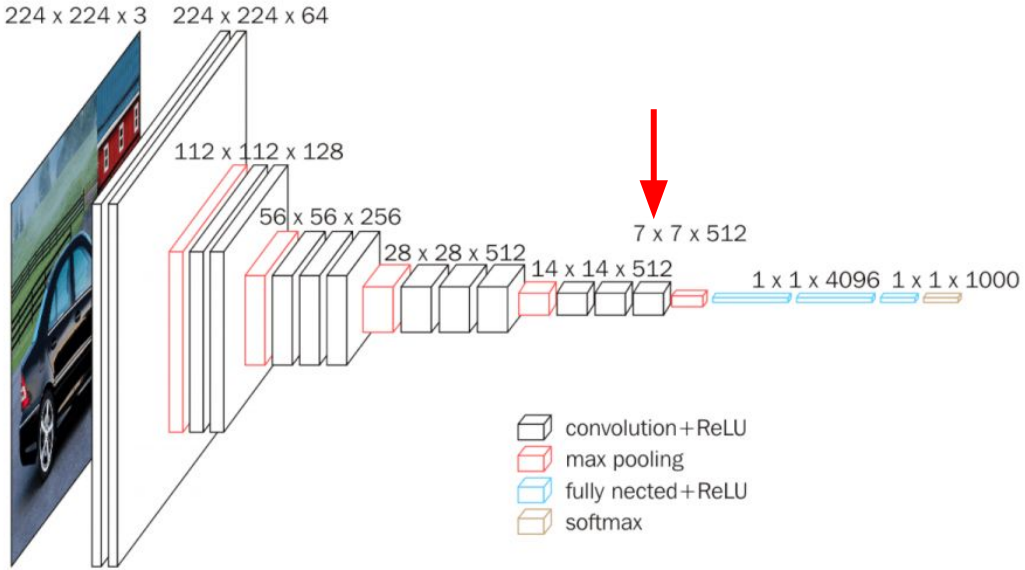
8. Здесь применяется 2-пулинг, размерность изображений уменьшается до 14x14.



Посчитаем количество весов в слоях VGG

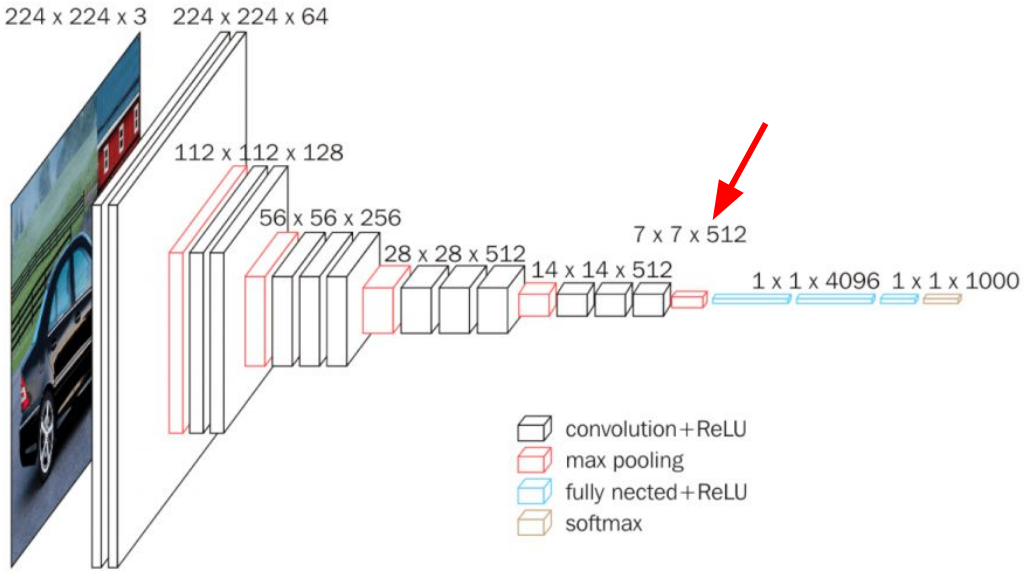
9. Применяем 512 фильтров размера 3x3,
и каждый фильтр 512-канальный.

Следовательно, тут 512*3*3*512=**2359296**
весов для тренировки.



Посчитаем количество весов в слоях VGG

8. Здесь применяется 2-пулинг, размерность изображений уменьшается до 7x7.



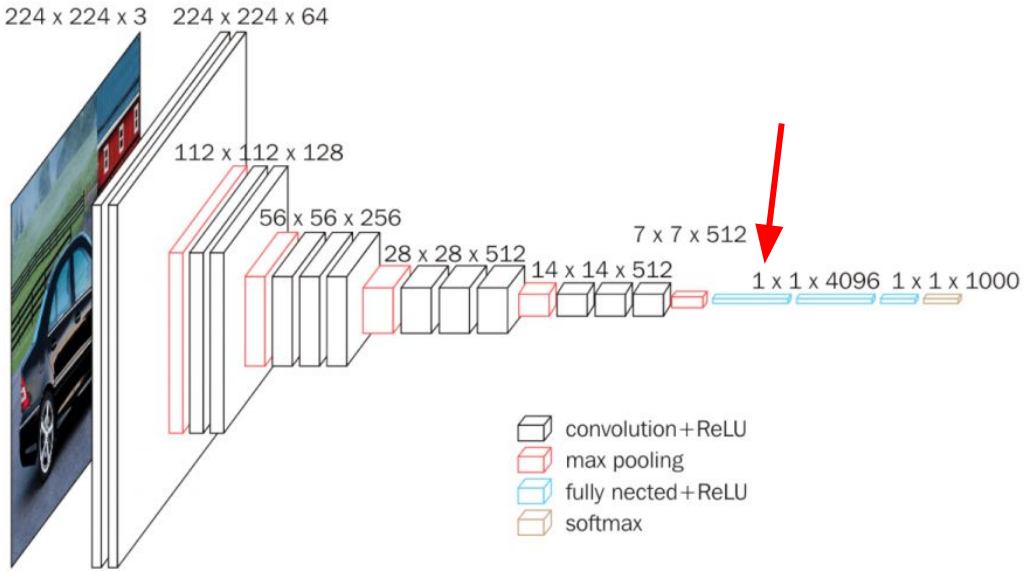
Посчитаем количество весов в слоях VGG

9. Здесь последний свёрточный слой соединяется с первым полносвязным слоем.

Имеем изображение 7x7 и 512 каналов. Т.е. слой будет состоять из 7*7*512=**25088** нейронов.

Следующий слой состоит из 4096 нейронов. Т.о. здесь будут 25088*4096=**102760448** весов-связей между нейронами.

Плюс еще **4096** весов-смещений в «голубом» слое.

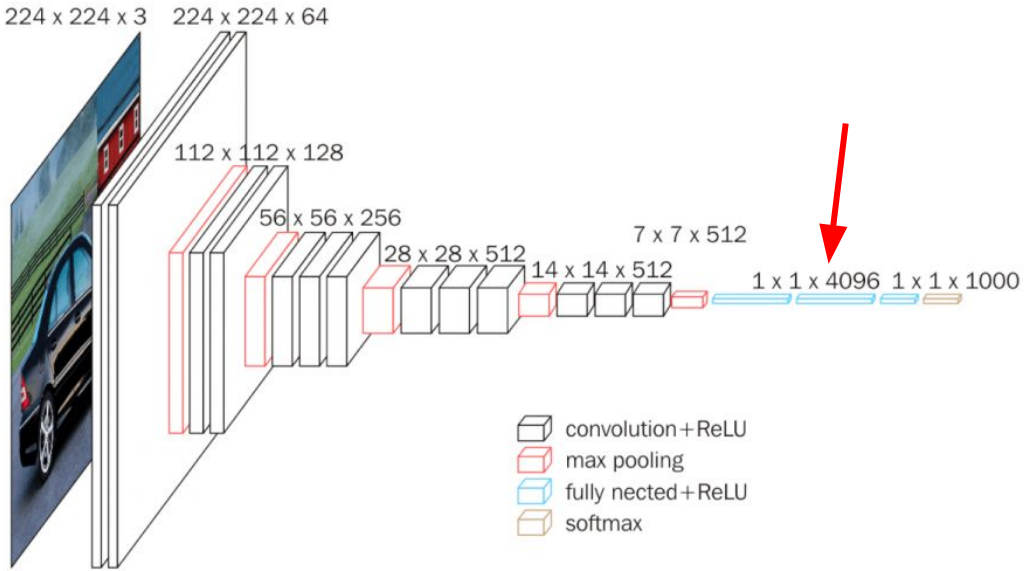


Посчитаем количество весов в слоях VGG

10. Здесь идёт второй полносвязный слой.

Т.о. здесь будут 4096*4096=**16777216** весов-связей между нейронами.

Плюс еще **4096** весов-смещений во втором «голубом» слое.

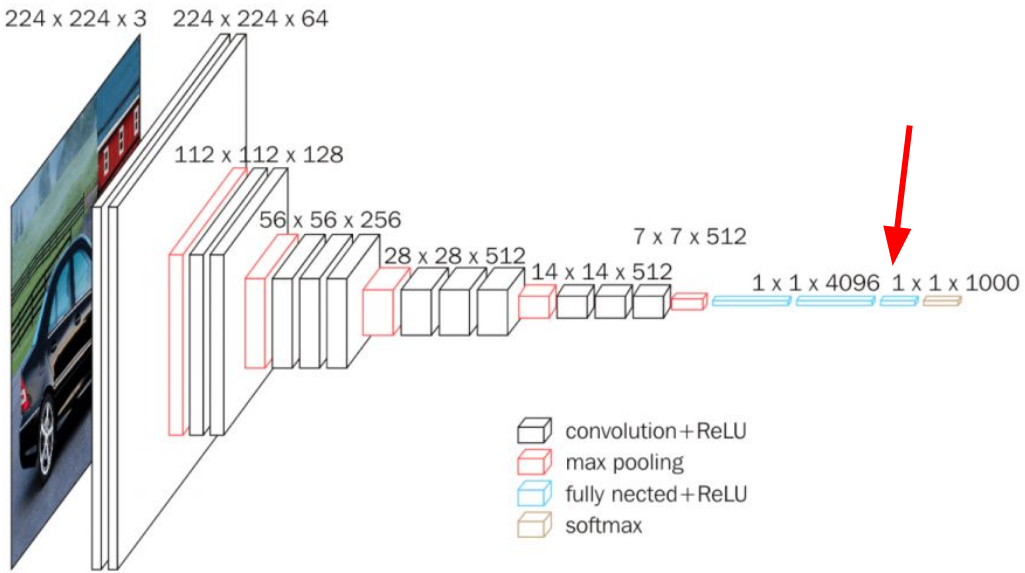


Посчитаем количество весов в слоях VGG

11. Здесь идёт третий полносвязный слой, связывающий группы из 4096 и 1000 нейронов.

Т.о. здесь будут 4096*1000=**4096000** весов-связей между нейронами.

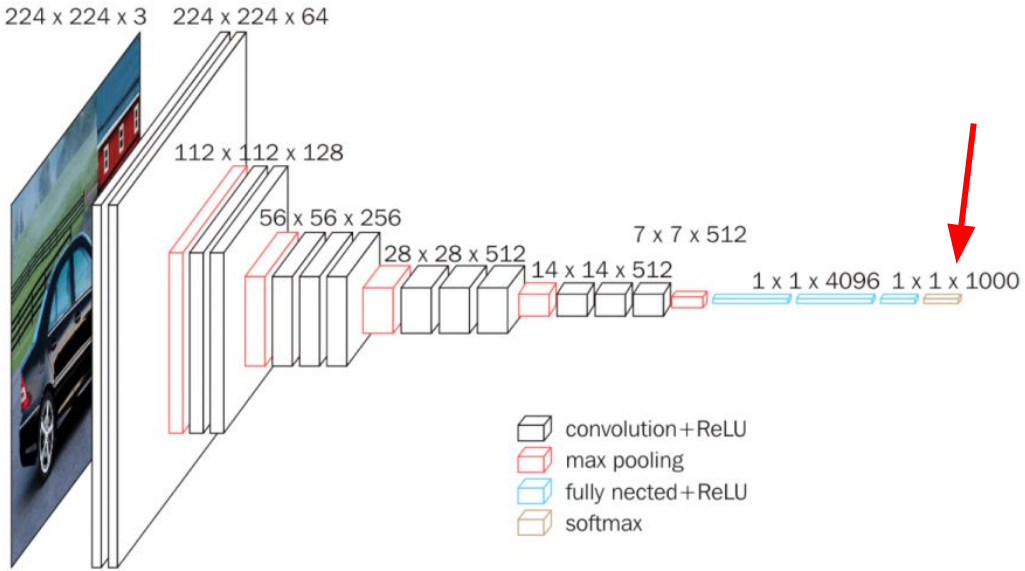
Плюс ещё **1000** весов-смещений в третьем «голубом» слое.



Посчитаем количество весов в слоях VGG

12. Потом идём слой softmax, не имеющий собственных весов.

Выходов у сети 1000 штук, поскольку она распознает картинки из 1000 классов.

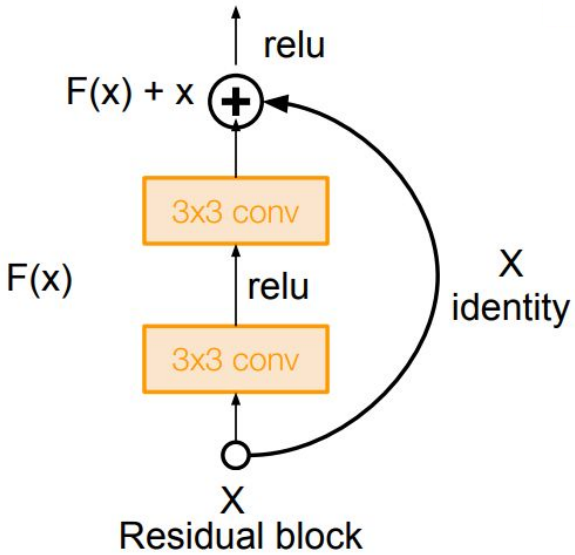


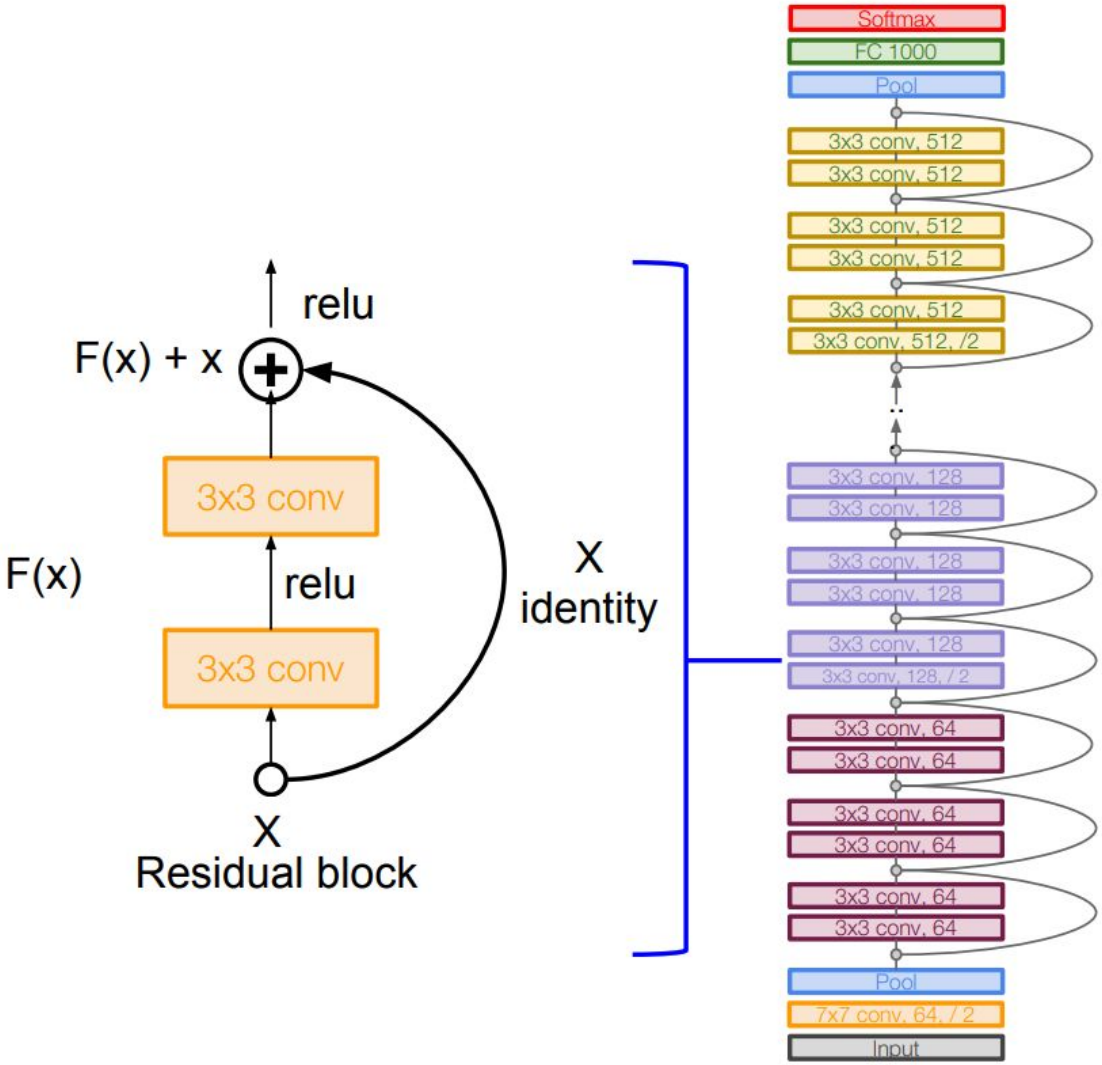
ResNet (2015)

ResNet

Её процент ошибочной классификации картинок из ImageNet 3.6% — ниже чем у человека)))

Особенность архитектуры: существуют прямые связи между далёкими слоями.




relu
F(x) + x
3x3 conv
F(x)
relu
3x3 conv
X
Residual block
X
identity
Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input

Принципы тренировки глубоких СНС

Тренировка СНС:

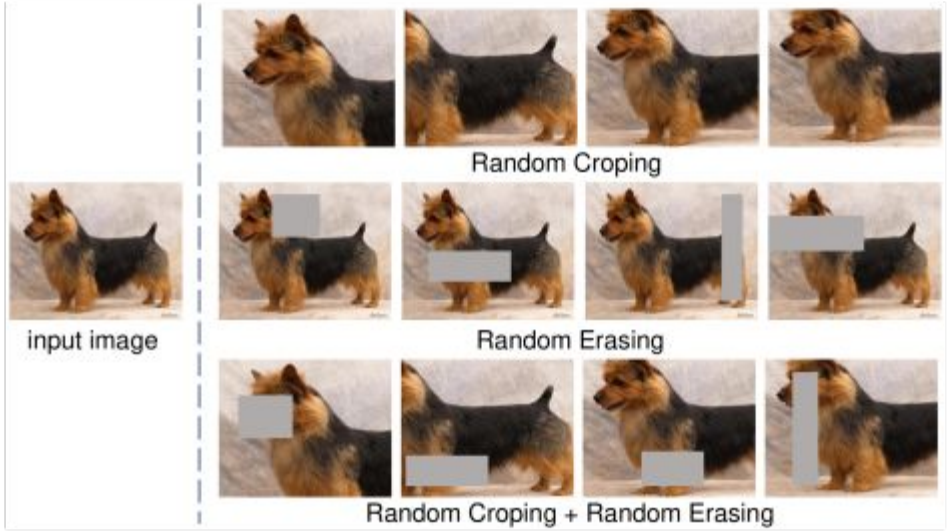
Взять тренировочную выборку (ТВ), построить функцию потерь, искать минимум функции потерь с помощью ГС (или его модификаций).

Обязательно использовать методы, препятствующие затуханию градиента: нормализация по мини-батчам, прямые связи между далёкими слоями (как в ResNet).

+ Особые методы подготовки ТВ (аугментация).
+ Особые методы подготовки новых картинок (не из ТВ) для распознавания (идея ансамбля).

Естественно, мы ниже предполагаем, что все изображения приведены к единому размеру перед отправкой в СНС.

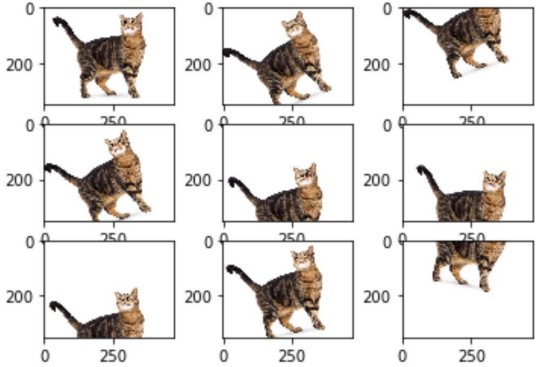
Аугментация



Была одна картинка из ТВ.

С помощью простых преобразований мы получаем несколько изображений для ТВ.

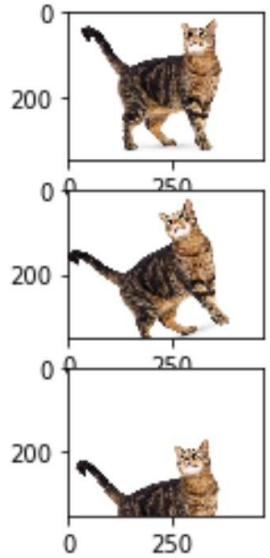
ТВ разбухает в несколько раз!



PROFIT!!!

Идея ансамбля

Пусть теперь эта картинка **не принадлежит** ТВ, и нам нужно её распознать (**СНС у нас уже натренирована**). Мы всё равно применяем к картинке аугментацию, получаем набор картинок:



Прогоняем все картинки (3 шт.) через СНС.
Получаем числа p_1, p_2, p_3 — это вероятность того, что соответствующая картинка принадлежит классу «кошка».

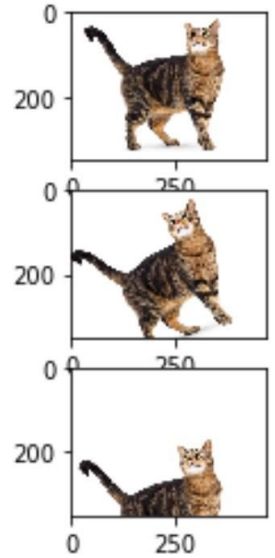
Что дальше можно сделать с этими числами? Ась?

Идея ансамбля

Получаем числа p_1, p_2, p_3 — это вероятность того, что соответствующая картинка принадлежит классу «кошка».

Что дальше делать с этими числами?

Ответ: нужно взять их среднее арифметическое $p=(p_1+p_2+p_3)/3$ и сказать, что число p — есть вероятность того, что на исходной картинке изображена кошка.



Transfer learning

Мотивация: лень

Сейчас есть огромные натренированные СНС (VGG, ResNet...) общего назначения (датасет ImageNet).
А вам нужно распознавать специфические изображения, которых нет в ImageNet (например, рентгеновские снимки или фото бактерий).

Что делать?
Тренировать свою СНС с нуля лень.

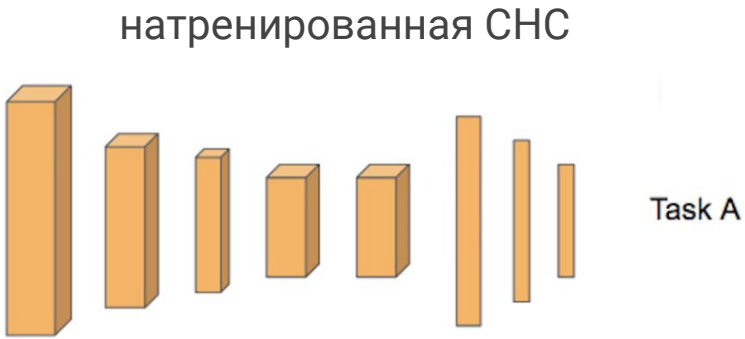
Решение: transfer learning.

Идея transfer learning

Берём любую натренированную СНС.
Её веса W имеют уже некоторые значения.

У нас есть своя ТВ. Составляем функцию потерь и начинаем её минимизировать с помощью ГС, **начиная со значений весов W**.

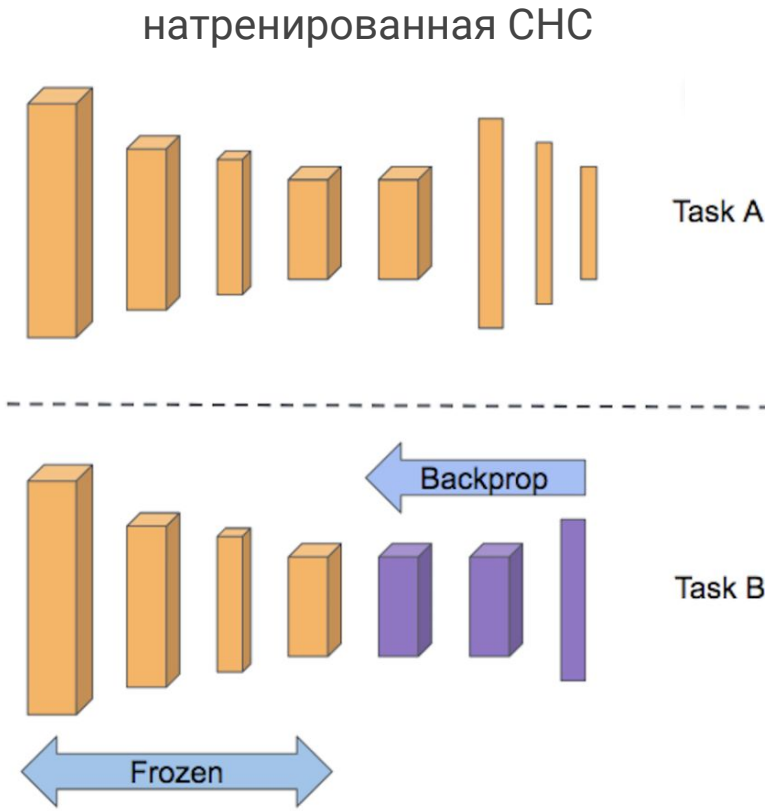
На каких весах ГС остановится — эти веса и будут оптимальными для вашей задачи.



Тонкая настройка transfer learning

Делать обычный ГС всё равно долго. Поэтому веса на первых слоях СНС «замораживают», то есть мы запрещаем ГС их изменять.

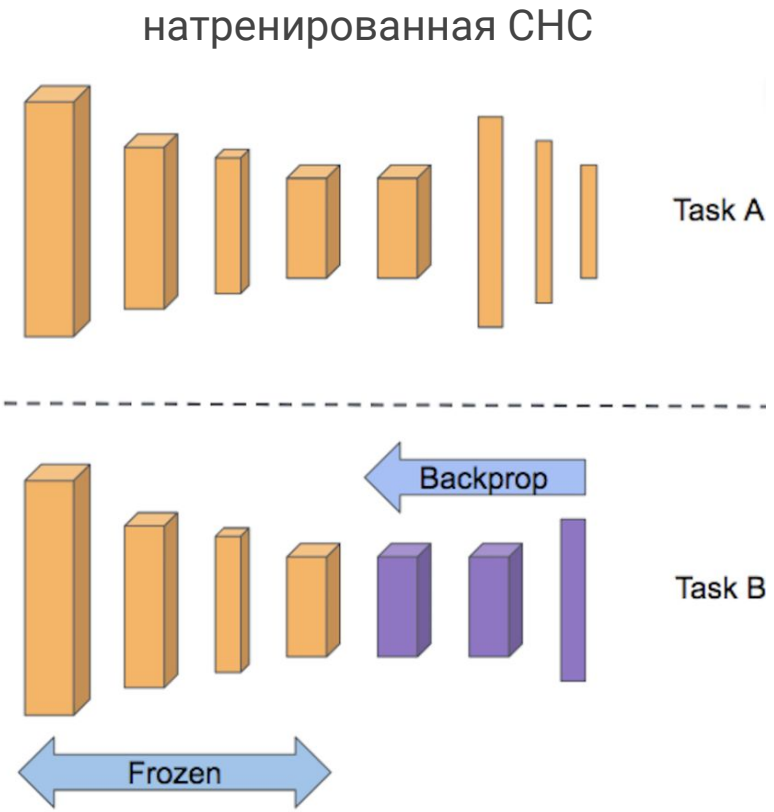
Смысл: веса на первых слоях СНС отвечают за самые основные свойства изображений, которые присутствуют в любой задаче по распознаванию картинок.



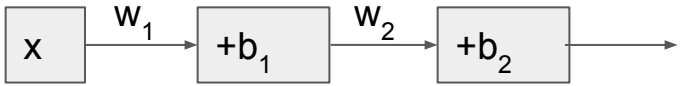
Тонкая настройка transfer learning

Более плавное решение:
не надо делать полностью «замороженных» слоёв. Нужно для каждого слоя S задать свой шаг h_S ГС. Чем ближе слой S к началу СНС, тем меньше его шаг h_S.

Фактически это означает, что веса из первых слоёв СНС будут меняться слабо.



Пример transfer learning (заморозка весов)



Дана НС (она не свёрточная), её функция $\mathbf{F_{NN}(x)=(w_1x+b_1)w_2+b_2}$.

Допустим, что её натренировали на некоторых данных и получили следующие **значения весов:** $w_1=1$, $b_1=1$, $w_2=1$, $b_2=-1$,
и НС стала вычислять функцию: $F_{NN}(x)=(1*x+1)*1-1=x$.
Будем дотренировывать эту НС на следующей ТВ:

X	Y
-1	1
2	4

Составим функцию потерь для этой ТВ:

$L(w)=((-w_1+b_1)w_2+b_2-1)^2+((2w_1+b_1)w_2+b_2-4)^2$

Находим ЧП:

$L(w)=((-w_1+b_1)w_2+b_2-1)^2+((2w_1+b_1)w_2+b_2-4)^2$

$\partial L/\partial w_1=-2w_2((-w_1+b_1)w_2+b_2-1)+4w_2((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial b_1=2w_2((-w_1+b_1)w_2+b_2-1)+2w_2((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial w_2=-2(-w_1+b_1)((-w_1+b_1)w_2+b_2-1)+2(2w_1+b_1)((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial b_2=2((-w_1+b_1)w_2+b_2-1)+2((2w_1+b_1)w_2+b_2-4).$

Стартовая точка для ГС: $w_1=1$, $b_1=1$, $w_2=1$, $b_2=-1$

По смыслу Transfer learning мы **заморозим** веса w_1,b_1,
а для остальных весов зададим шаг h=0.1

Вычислим значения ЧП:

$\partial L/\partial w_1(1,1,1,-1)=-4$ $\partial L/\partial b_1(1,1,1,-1)=-8$

$\partial L/\partial w_2(1,1,1,-1)=-12,$ $\partial L/\partial b_2(1,1,1,-1)=-8$

$\partial L/\partial w_1(1,1,1,-1)=-4$ $\partial L/\partial b_1(1,1,1,-1)=-8$

$\partial L/\partial w_2(1,1,1,-1)=-12,$ $\partial L/\partial b_2(1,1,1,-1)=-8$

Поскольку веса w_1, b_1 заморожены, то ГС преобразует только веса w_2, b_2 :

$w_2:=w_2-h*\partial L/\partial w_2(1,1,1,-1)=1-0.1*(-12)=2.2$

$b_2:=b_2-h*\partial L/\partial b_2(1,1,1,-1)=-1-0.1*(-8)=-0.2$

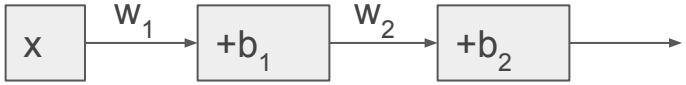
Таким образом, после одной итерации ГС веса НС станут:

$w_1=1, b_1=1, w_2=2.2, b_2=-0.2$

и теперь НС вычисляет функцию:

$F_{NN}(x)=(1*x+1)*2.2-0.2=2.2x+2$

Transfer learning (шаг ГС зависит от слоя)



Дана НС (она не свёрточная), её функция $\mathbf{F_{NN}(x)=(w_1x+b_1)w_2+b_2}$.

Допустим, что её натренировали на некоторых данных и получили следующие **значения весов:** $w_1=1$, $b_1=1$, $w_2=1$, $b_2=-1$, и НС стала вычислять функцию: $F_{NN}(x)=(1*x+1)*1-1=x$.

Будем дотренировывать эту НС на следующей ТВ:

X	Y
-1	1
2	4

Составим функцию потерь для этой ТВ:

$$L(w)=((-w_1+b_1)w_2+b_2-1)^2+((2w_1+b_1)w_2+b_2-4)^2$$

Находим ЧП:

$L(w)=((-w_1+b_1)w_2+b_2-1)^2+((2w_1+b_1)w_2+b_2-4)^2$

$\partial L/\partial w_1=-2w_2((-w_1+b_1)w_2+b_2-1)+4w_2((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial b_1=2w_2((-w_1+b_1)w_2+b_2-1)+2w_2((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial w_2=-2(-w_1+b_1)((-w_1+b_1)w_2+b_2-1)+2(2w_1+b_1)((2w_1+b_1)w_2+b_2-4),$

$\partial L/\partial b_2=2((-w_1+b_1)w_2+b_2-1)+2((2w_1+b_1)w_2+b_2-4).$

Стартовая точка для ГС: $w_1=1, b_1=1, w_2=1, b_2=-1$

Установим **разный шаг для разных слоёв.**
Для весов w_1,b_1 шаг равен $h_1=0.05$,
а для остальных весов зададим шаг $h_2=0.1$

Вычислим значения ЧП:

$\partial L/\partial w_1(1,1,1,-1)=-4$ $\partial L/\partial b_1(1,1,1,-1)=-8$

$\partial L/\partial w_2(1,1,1,-1)=-12,$ $\partial L/\partial b_2(1,1,1,-1)=-8$

$\partial L/\partial w_1(1,1,1,-1)=-4$ $\partial L/\partial b_1(1,1,1,-1)=-8$

$\partial L/\partial w_2(1,1,1,-1)=-12,$ $\partial L/\partial b_2(1,1,1,-1)=-8$

ГС получает новые значения весов:

$w_1:=w_1-h_1*\partial L/\partial w_1(1,1,1,-1)=1-0.05*(-4)=1.2$

$b_1:=b_1-h_1*\partial L/\partial b_1(1,1,1,-1)=1-0.05*(-8)=1.4$

$w_2:=w_2-h_2*\partial L/\partial w_2(1,1,1,-1)=1-0.1*(-12)=2.2$

$b_2:=b_2-h_2*\partial L/\partial b_2(1,1,1,-1)=-1-0.1*(-8)=-0.2$

Таким образом, после одной итерации ГС веса НС станут:

$w_1=1.2$, $b_1=1.4$, $w_2=2.2$, $b_2=-0.2$

и теперь **НС вычисляет функцию:**

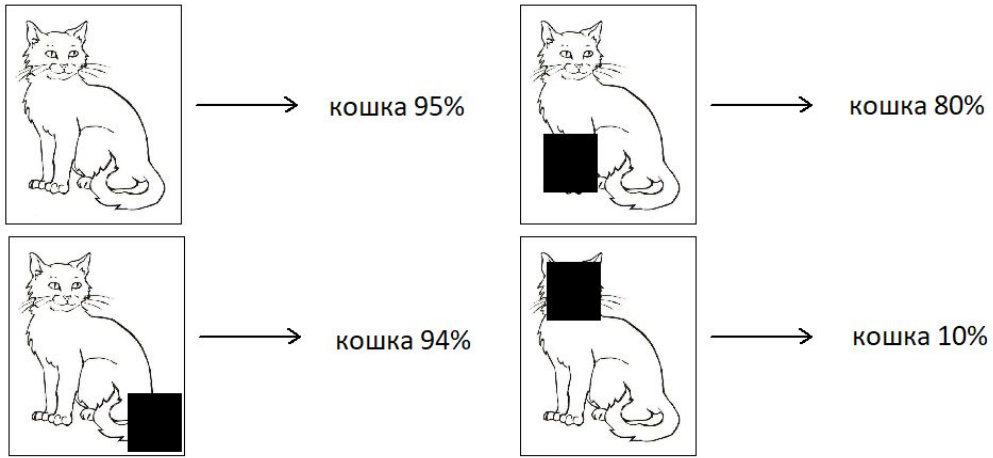
$F_{NN}(x)=(1.2*x+1.4)*2.2-0.2=2.64x+2.88$

СНС действительно понимает, что видит?

СНС действительно видит то, что мы ожидаем от неё?

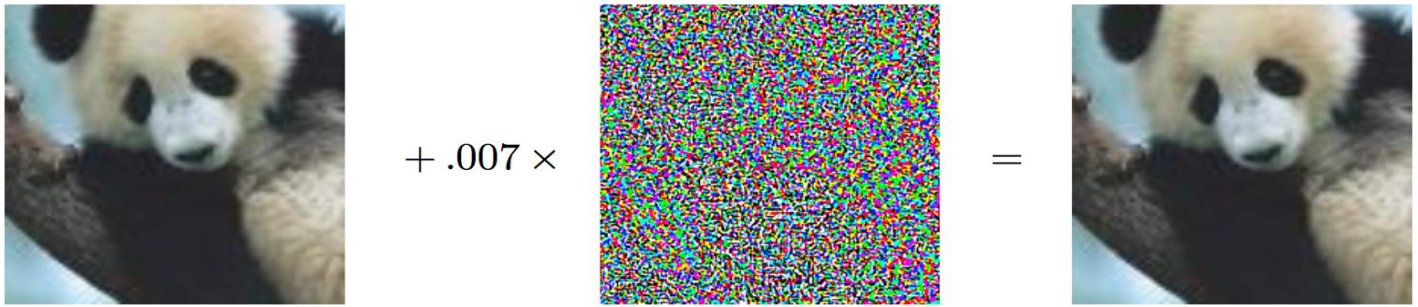
Это очень важный вопрос!
Если СНС анализирует совершенно другие признаки изображения (чем человек), то применять такие СНС на практике рискованно.

Можно подбором найти области изображения, которые сеть считает существенными:

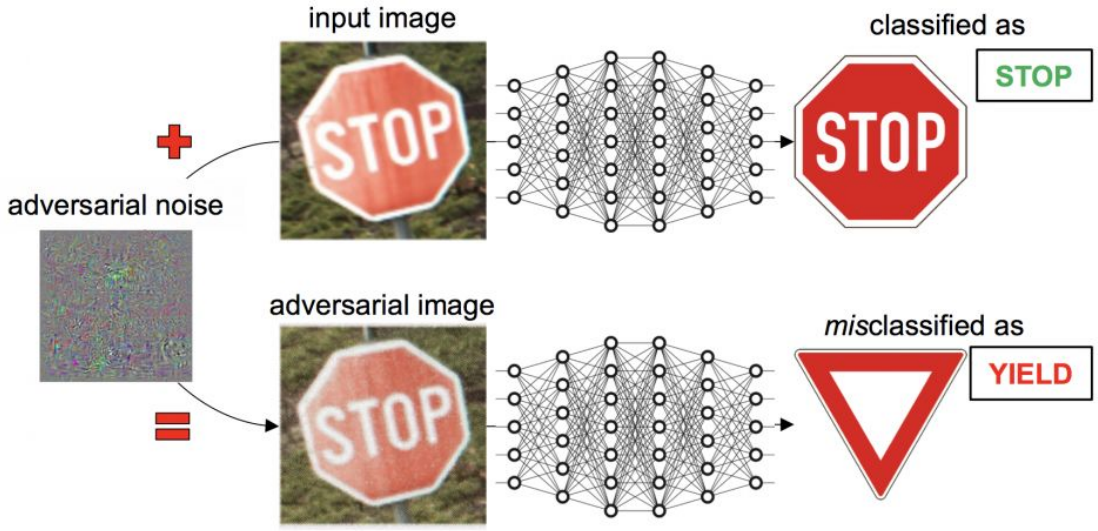


Как обмануть CHC? Adversarial attack — это…

…когда мы добавляем незаметный (для человеческого глаза) шум в изображение, и CHC начинает его неправильно классифицировать.



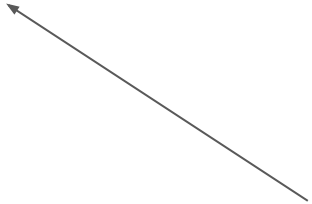
Ещё более жестокий пример:



Можно не добавлять шум, а приклеить наклейки на фотографируемый предмет.

Учебная задача (очень упрощённо)

Будем тренировать СНС, которая должна научиться распознавать флаги стран. Для простоты будем учиться отличать флаг СССР от флага Польши.



Обратите внимание, что оттенки красного на флагах разные.

*- флаги выбраны не из-за политических предпочтений, а из-за простоты рисунка.

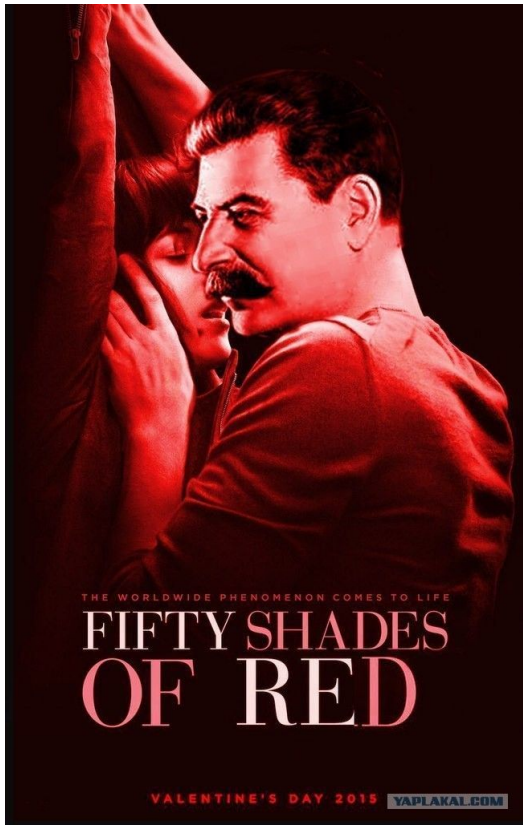
Формат данных на вход CHC

Поскольку наши флаги состоят из горизонтальных полос (и число полос не более 2), то наши данные по сути двухпиксельные:

p_1
p_2

Поскольку на вход CHC будут подаваться лишь оттенки красного, то будем использовать лишь координату R в формате RGB.

То есть вместо тройки (150,0,0) будем писать просто 150.

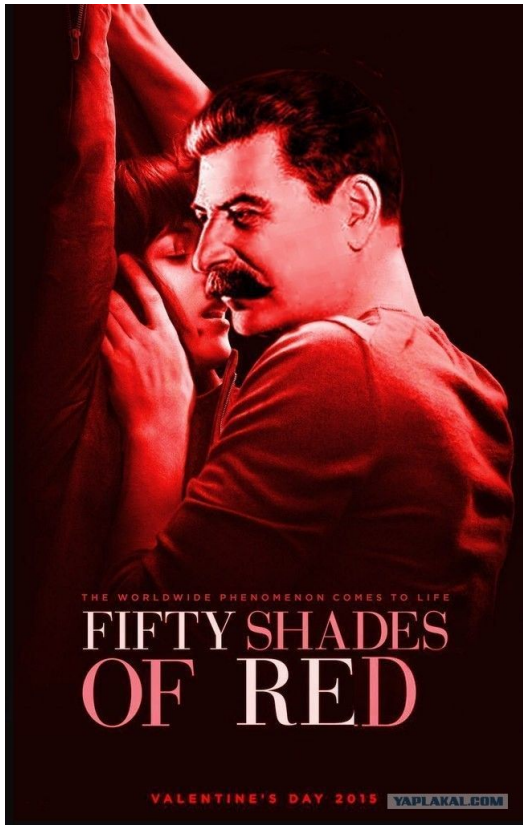


Формат данных на вход СНС

Таким образом, пиксели p_1,p_2 принимают значение от 0 (белый цвет) до 255 (ярко-красный).

p_1
p_2

Мы предполагаем, что наша ТВ содержит флаги с разными оттенками красного (имейте в виду, что белая полоса на польском флаге фактически может быть сильно разбавленным красным).



Задача распознавания не такая уж простая

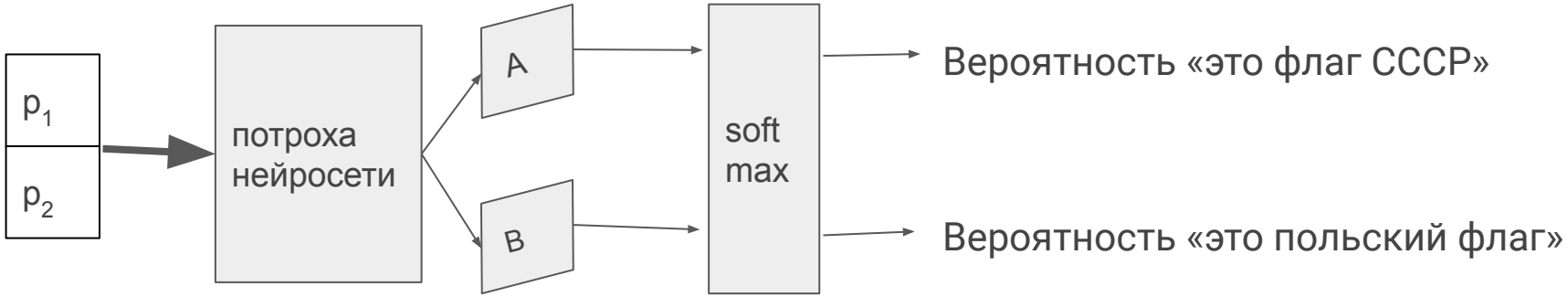
Дело в том, что

- флаг СССР не обязательно имеет все пиксели одного цвета;
- флаг Польши не обязательно содержит чистый белый цвет. Фактически верхняя полоса может быть бледно-светло-красной.

Это означает, что значимые тут закономерности такие:

- **пиксели на флаге СССР должны быть как можно краснее;**
- **нижний пиксель на польском флаге как можно краснее, а верхний пиксель как можно белее.**

Что выдаст СНС после тренировки?

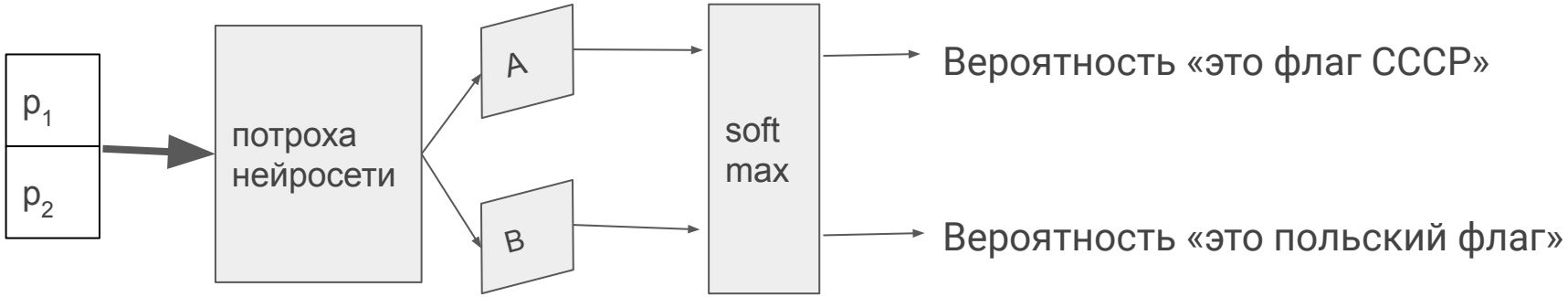


Вероятности пропорциональны значениям A,B. Используя закономерности

- **пиксели на флаге СССР должны быть как можно краснее;**
- **нижний пиксель на польском флаге как можно краснее, а верхний пиксель как можно белее.**

получаем, что выражения A,B имеют вид...

Что выдаст СНС после тренировки?

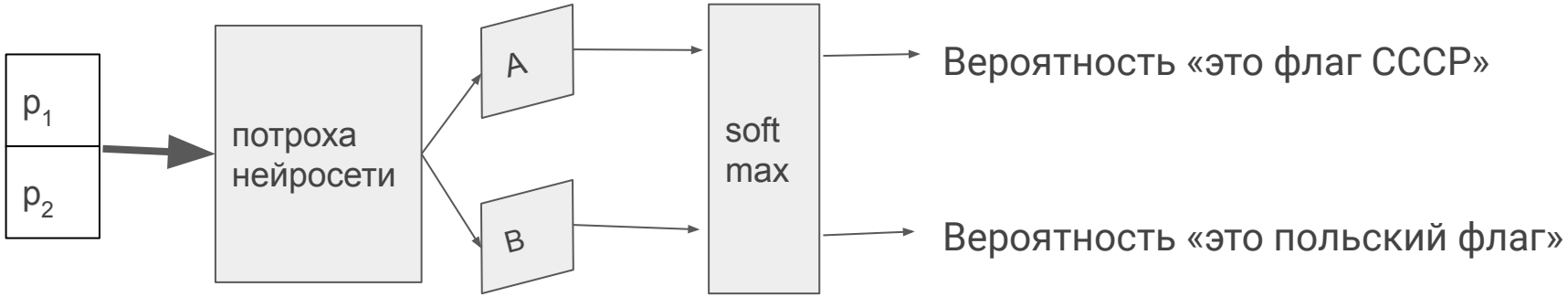


$A=...+ap_1+bp_2+...$ где числа a,b,c,d — положительные

$B=...-cp_1+dp_2+...$

Какие есть **соотношения между числами** a,b,c,d?

Что выдаст СНС после тренировки?



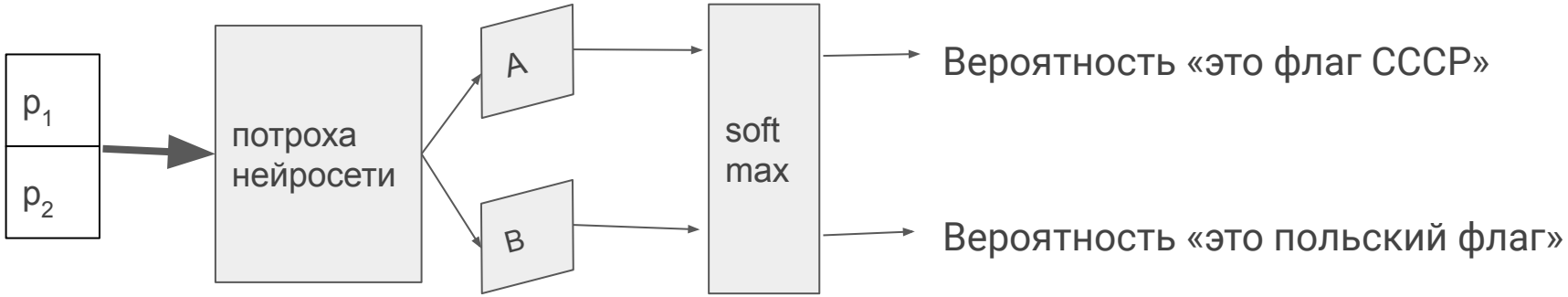
$A=...+ap_1+bp_2+...$ где числа a,b,c,d — положительные

$B=...-cp_1+dp_2+...$

Очевидно, что если верхняя полоса белая ($p_1=0$), то это должен быть флаг Польши. Следовательно, нужно потребовать A<B, и поэтому **d>b**.

Поехали дальше...

Что выдаст СНС после тренировки?



$A=...+ap_1+bp_2+...$ где числа a,b,c,d — положительные, **d>b**.

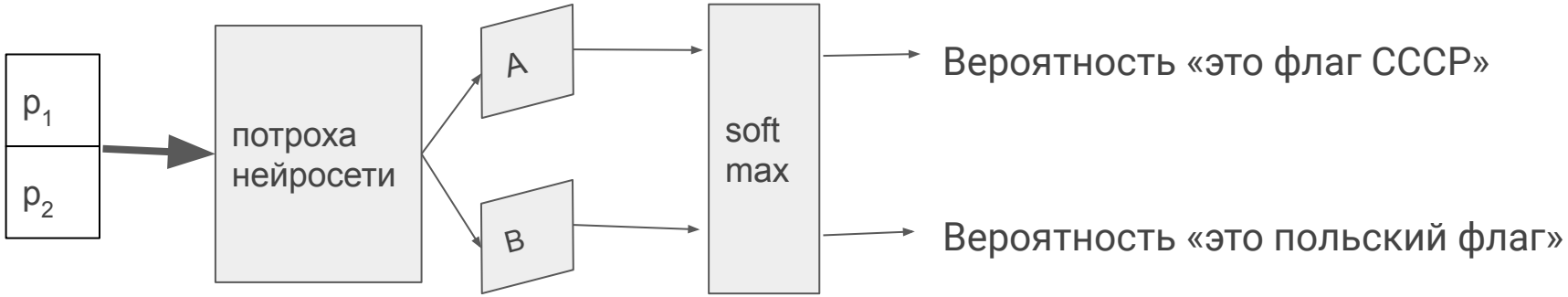
$B=...-cp_1+dp_2+...$

Составим неравенство: A<B (скорее Польша, чем СССР),

$ap_1+bp_2<-cp_1+dp_2$ $p_2>p_1(a+c)/(d-b)$

знак последнего неравенства именно такой, поскольку **d>b**.

Что выдаст СНС после тренировки?



$A=...+ap_1+bp_2+...$ $B=...-cp_1+dp_2+...$

Будет Польша, если $p_2>p_1(a+c)/(d-b)$

Как с помощью этой формулы обмануть при этом нейросеть (человека)?

А тут нужно использовать **особенности человеческого восприятия** (это и обусловливает расхождение ответов человека и СНС в последующих примерах).

Особенность восприятия цветов

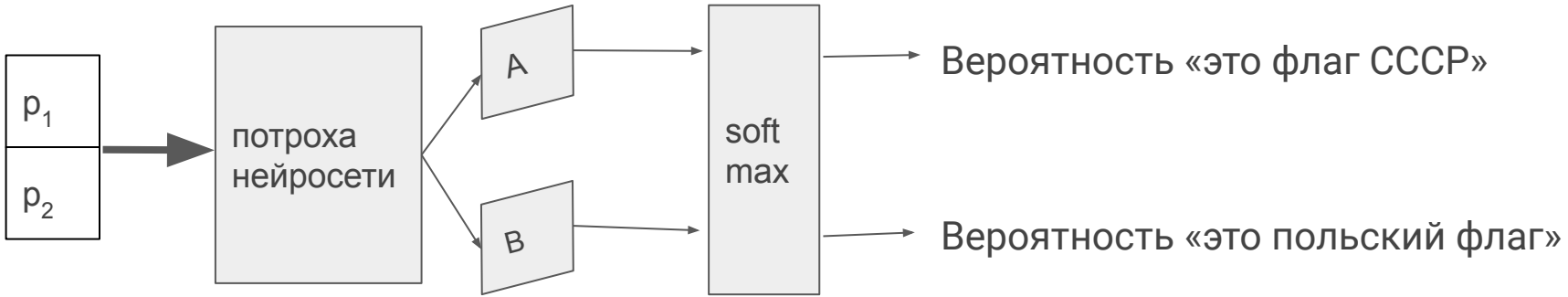
Человек по-разному реагирует на разную интенсивность света (цвета). Ему сложно различать цвета слишком ярких объектов (солнце, сильный огонь итд).

Как следует из этой таблицы, человек визуально не различает предметы с температурой >1300 градусов.

	Цвет накала	Температура, °C
	Тёмнокоричневый	530-580
	Коричнево-красный	580-650
	Тёмнокрасный	650-730
	Тёмновишнёво-красный	730-770
	Вишнёво-красный	770-800
	Светловишнёво-красный	800-830
	Светлокрасный	830-900
	Оранжевый	900-1050
	Тёмножёлтый	1050-1150
	Светложёлтый	1150-1250
	Ослепительно-белый	1250-1300

На этом и будет основан обман))))

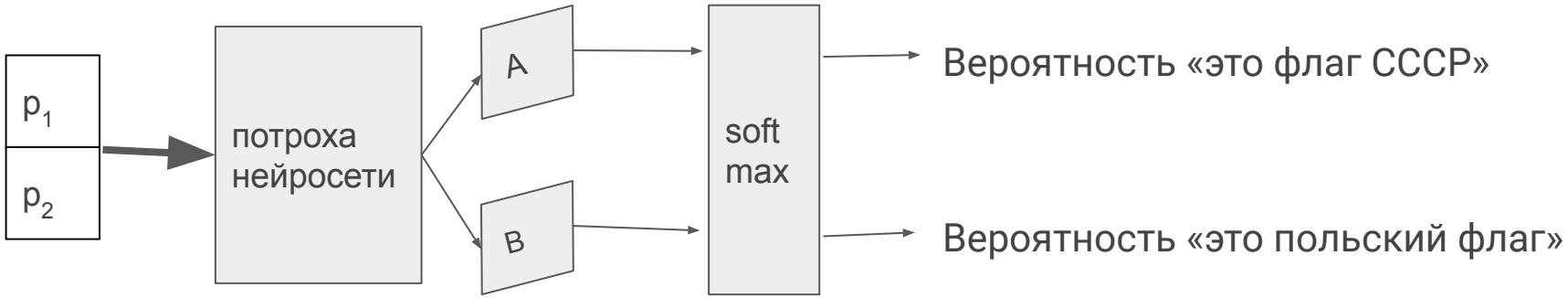
Нужно просто взять цвета с «запредельной интенсивностью».
Можно даже не из диапазона [0,255].



$A=...+ap_1+bp_2+...$ $B=...-cp_1+dp_2+...$ Будет Польша, если $p_2>p_1(a+c)/(d-b)$

Например, для $(a+c)/(d-b)=2$ будет Польша, если $p_2>2p_1$

На этом и будет основан обман))))



Будет Польша, если $p_2 > 2p_1$

Запустим на вход СНС пару чисел (255,512) — СНС классифицирует это как Польшу. Но для человека (в силу ограниченности его органов чувств) красный с интенсивностью 512 **не отличим** от красного с интенсивностью 255.

Человек на паре (255,512) увидит:

А ИИ будет считать, что Польша)))

Выводы

- Мы познакомились с некоторыми известными большими и натренированными СНС.

- Изучили технологию transfer learning, позволяющую дотренировывать СНС общего назначения для решения частной задачи.

- Познакомились с процессом аугментации ТВ.

- Рассмотрели основные способы внешних атак на СНС.