Ускорение инференса

Константин Архипенко

9 декабря 2020 г.

Мотивация

ИСП РАН

Нейронные сети постепенно проникают повсюду и скоро доберутся до микроволновок:

- Автопилот в автомобиле: в реальном времени обрабатываем видео
- В телефонах фото автоматически корректируются сетями

Потребности

- Работать быстро
- Использовать поменьше ресурсов (памяти)

Как ускоряться (план лекции)

ИСП РАН

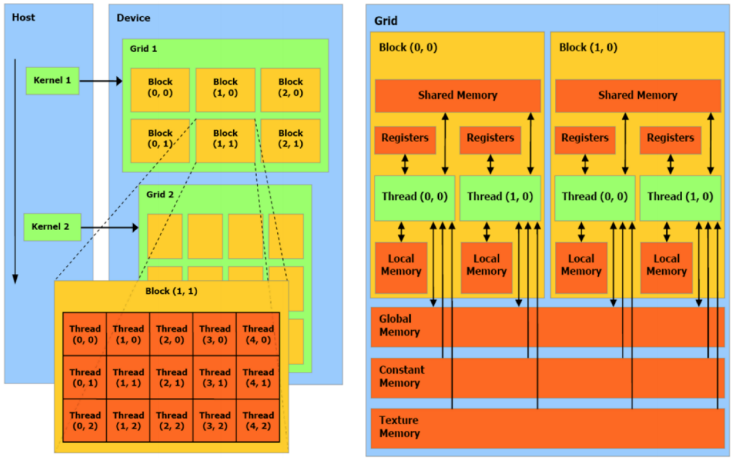
Ускорять *уже обученную* модель можно за счет

1. специализированного железа (*оптимизированные операции*, *квантизация*)
2. удаления ненужных нейронов (*pruning*)

Также можно

3. придумывать заведомо эффективные архитектуры
   - depthwise separable, grouped свертки (SqueezeNet, MobileNet и др.). Рассматривалось на лекции 5
4. переносить знания с большой модели на меньшую (*дистилляция знаний*)

- NVIDIA CUDA


Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)
Grid
Block (0, 0)
Block (1, 0)
Shared Memory
Registers
Thread (0, 0)
Thread (1, 0)
Local Memory
Global Memory
Constant Memory
Texture Memory

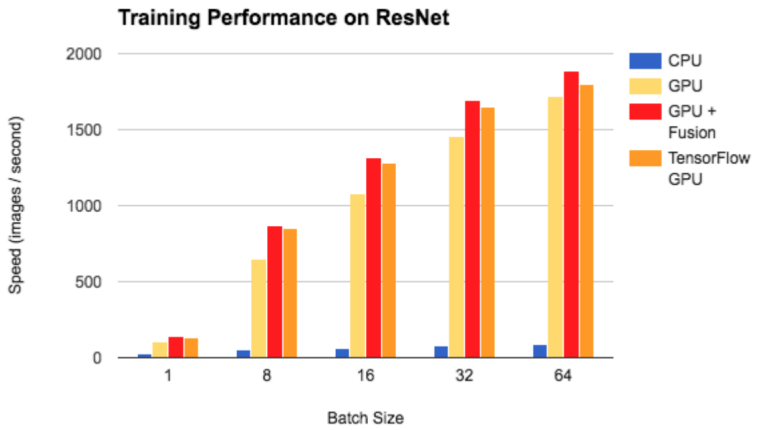
Kernel fusion

ИСП РАН

Пример: $\alpha x + \beta$

- Kernel 1: загрузить элемент вектора x из глобальной памяти, умножить на α, вернуть результат в глобальную память
- Kernel 2: загрузить элемент вектора αx из глобальной памяти, прибавить β, вернуть результат в глобальную память
- Объединить в один kernel — эффективнее

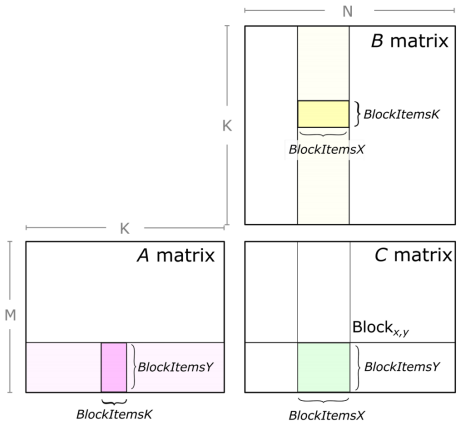
Kernel fusion



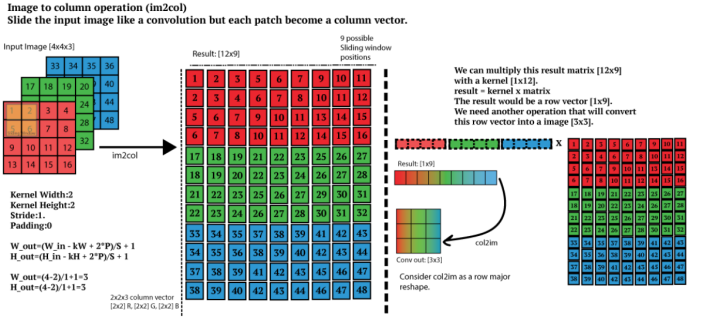
Умножение матриц (GEMM)

ИСП РАН

- Blocking (ускоряемся благодаря кэшу)
- SIMD для CPU (SSE, AVX)



im2col (чтобы применить GEMM к сверткам)



Winograd convolution

ИСП РАН

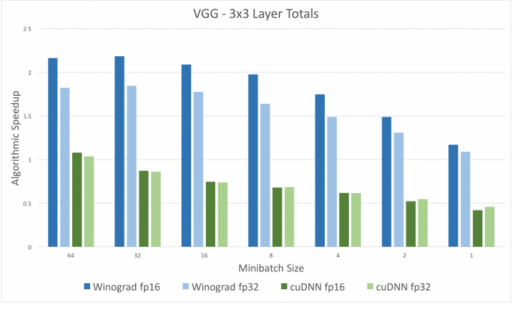
6 multiplications

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_0 + m_1 + m_2 \\ m_1 - m_2 - m_3 \end{bmatrix}$$

$$m_0 = (d_0 - d_2)g_0 \qquad m_1 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2}$$

$$m_2 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2} \qquad m_3 = (d_1 - d_3)g_2$$

4 multiplications



Winograd convolution

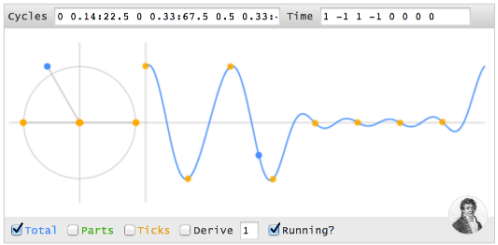
ИСП РАН

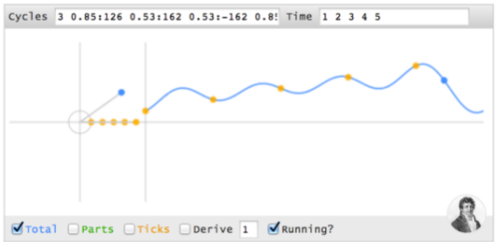
- Искать такие эффективные умножения (с меньшим числом умножений скаляров) очень непросто
- Один из вариантов — представить свертку как произведение полиномов (у которых коэффициенты соответствуют входу и фильтру) и сделать интерполяцию по значениям в нескольких точках
- Как выбирать множество точек — опять же искусство

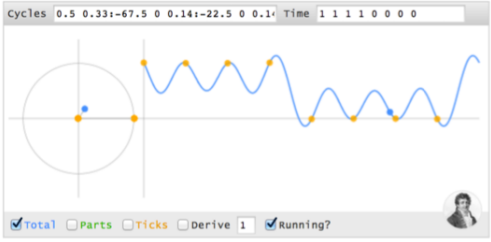
Fast Fourier convolution

ИСП РАН

- Дискретное преобразование Фурье (image credit: betterexplained.com)







$$X_k = \sum_{n=0}^{N-1} x_n \cdot \exp(-2\pi i k n / N)$$

- С помощью несложного трюка считается рекурсивно за $O(N \log N)$
- Обратное преобразование:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot \exp(2\pi i k n / N)$$

- Можем ускориться за счет свойства: Фурье-образ результата свертки — *произведение* Фурье-образов входа и фильтра

Network pruning

ИСП РАН

- Основная идея — убрать ненужные веса/нейроны, тем самым уменьшив число весов и приближаясь к вычислениям над разреженными матрицами
- Самый простой подход — зануление малых весов по порогу при обучении
- Можно занулять за отсутствие вклада в score на валидации, за похожесть весов и т. д.
- Еще подход — обучение с l_1 и l_2 регуляризацией для “натурального” зануления части весов

Квантизация (quantization)

ИСП РАН

- Один из подходов: Jacob et al. (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference / *CVPR 2018*

Переводим все операции в целочисленные

$$\mathbf{r} = S(\mathbf{q} - Z)$$

- $r^{(i)}, s \in \text{fp32}$, $q^{(i)}, Z \in \text{uint8}$
- S, Z — подбираемые во время обучения скаляры, *общие для всего слоя*

По определению произведения матриц

$$\mathbf{r}_3 = \mathbf{r}_1 \mathbf{r}_2$$

$$q_3^{(i,k)} = Z_3 + \frac{S_1 S_2}{S_3} \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2)$$

- На практике $M = \frac{S_1 S_2}{S_3} \in (0, 1)$ и поэтому представимо как:

$$M = 2^{-n} M_0 \quad M_0 \in [0.5, 1)$$

- Fixed-point умножение (

ИСП РАН

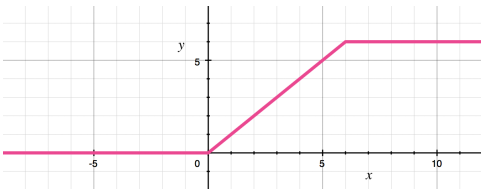
Перепишем еще раз

$$q_3^{(i,k)} = Z_3 + M(NZ_1Z_2 - Z_1a_2^{(k)} - Z_2\bar{a}_1^{(i)} + \sum_{j=1}^{N} q_1^{(i,j)}q_2^{(j,k)})$$

- $q_1^{(i,j)}q_2^{(j,k)} \in$ uint16
- Все слагаемые в скобках $\in$ int32
- Для представления M_0 также используем 32 бита
- Правую часть приводим к uint8 (*saturating cast*)

Магическая “6”

- Используется для обучения нейросетей, которые потом планируется квантизовать
- Оказалось, что для 8-битной квантизации лучше подходит “6”
- После квантизации роль функции активации играет как раз saturating cast



Прибавим bias к произведению $q_1 q_2$

ИСП РАН

- Но bias уже будет не uint8, как вход слоя $\mathbf{q}_1$ и веса $\mathbf{q}_2$, а int32 с фиксированными $S_{\text{bias}} = S_1 S_2$ и $Z_{\text{bias}} = 0$
- На практике для bias нужна хорошая точность, так как он прибавляется к признакам всех объектов
- Получаем fused слой (matmul/conv + bias + cast в роли функции активации)

Fake quantization во время обучения

ИСП РАН

- Используем промежуток квантизации $[a, b]$ и $n = 2^8$ уровней

Fake quantization

$$\text{clamp}(r; a, b) = \min(\max(x, a), b) \qquad s(a, b, n) = \frac{b - a}{n - 1}$$

$$q(r; a, b, n) = \text{round}\Big(\frac{clamp(r; a, b) - a}{s(a, b, n)}\Big)s(a, b, n) + a$$

Как оценить a и b?

- Для матриц весов: $a := \min w$, $b := \max w$
- Для выходов ReLU6: оцениваем с помощью EMA

Подбор S и Z для слоев

ИСП РАН

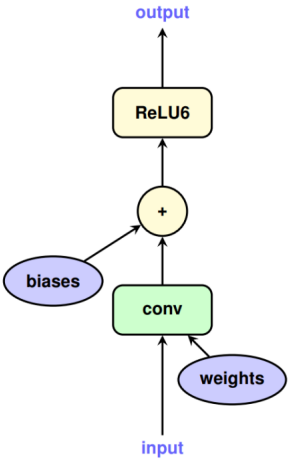
- После обучения чуть сдвигаем промежуток $[a, b]$, чтобы 0.0 точно соответствовал uint8-числу $z(a, b, n)$
- Тогда:

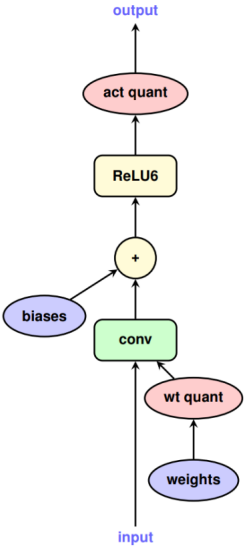
$$S := s(a, b, n) \qquad Z := z(a, b, n)$$

Преимущество обучения с fake quantization

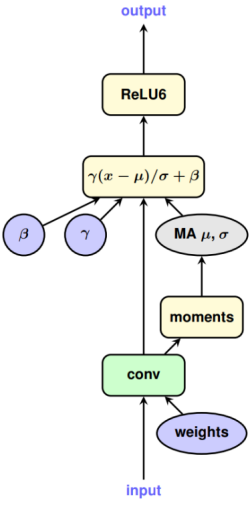
- Меньше падает качество по сравнению с квантизацией pure float модели

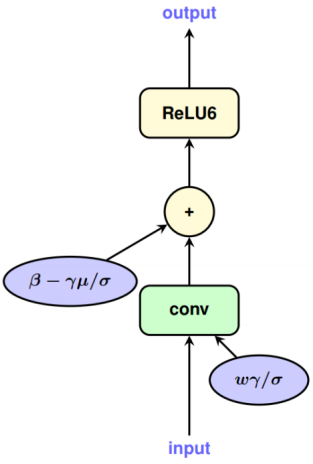
Обучения слоя: обычное и с fake quantization





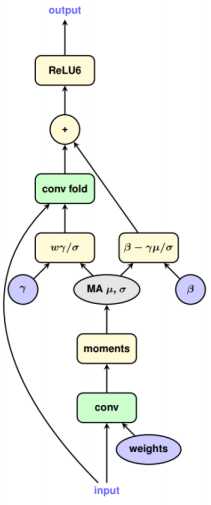
Слой с батч-нормализацией

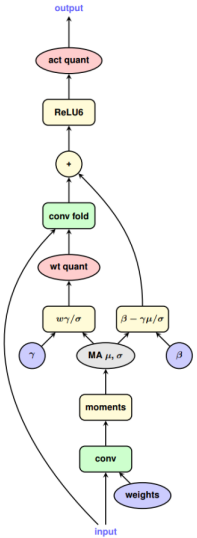




Fake quantization для слоя с батч-нормализацией

ИСП РАН



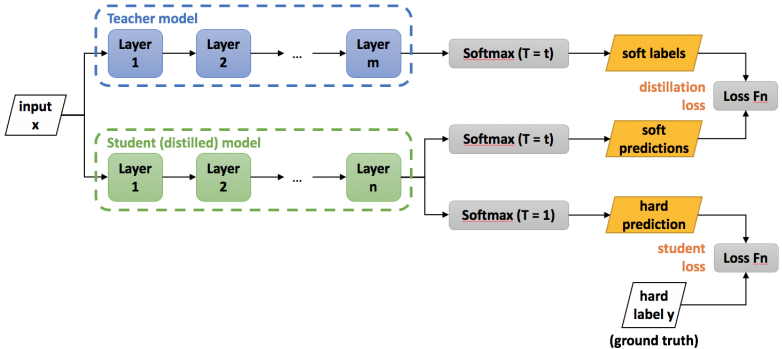


Дистилляция знаний (knowledge distillation)

ИСП РАН

Цель

- Используя большую сеть, обучить малую сеть, чтобы она имела аналогичное качество на целевой задаче
- Сеть-“студент” мимикрирует под выход сети-“учителя”



Softmax with Temperature

ИСП РАН

- Вспомним softmax:

$$\text{softmax}_i(\mathbf{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \in [0, 1] \qquad \sum_i \text{softmax}_i(\mathbf{z}) = 1$$

Проблема

- Слишком хороший учитель будет выдавать близкую к 1 вероятность для правильного класса и нули для остальных
- Между тем полезными знаниями для студента могли бы быть связи между классами (напр. какая цифра больше всего похожа на единицу?)

ИСП РАН

Вводим *температуру* (Hinton et al. 2015)

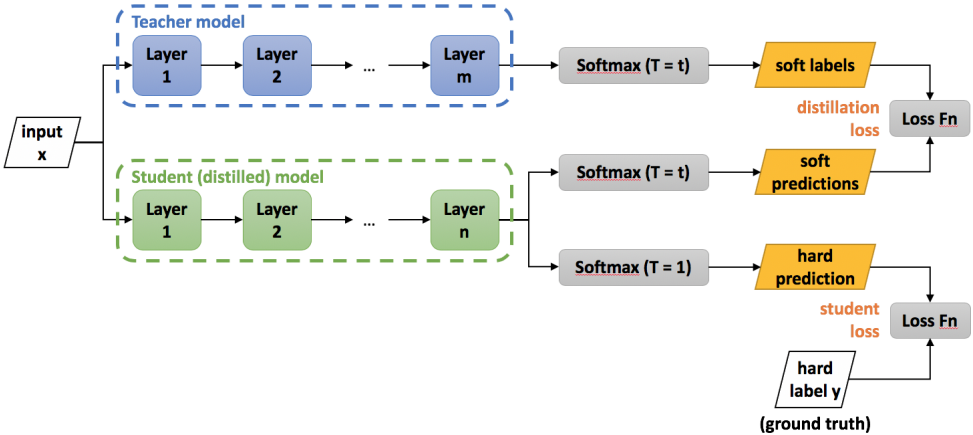
$$\text{softmax}_i(\mathbf{z}, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \qquad T > 1$$

- Вводим *distillation loss* (насколько хорошо студент “мимикрирует”), в котором выходы учителя с $\text{softmax}(\cdot, T)$ используются как soft labels
- Комбинируем с лоссом студента на реальных метках и с обычным softmax (важность того или иного лосса можно варьировать):

$$L = \alpha L_{\text{dist}} + (1 - \alpha) L_{\text{stud}} \qquad \alpha \in (0, 1]$$

Схема из статьи Hinton et al. 2015

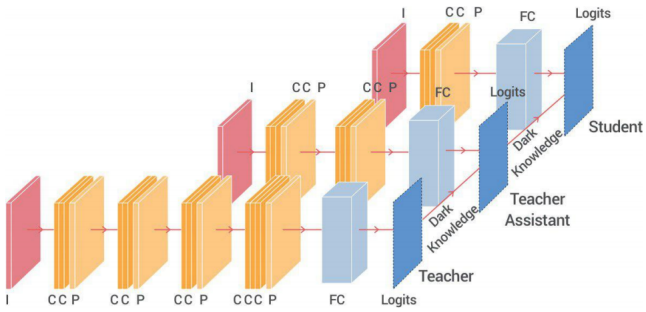
ИСП РАН



ИСП РАН

Teacher assistant (промежуточный учитель), Mirzadeh et al. 2019

- Может быть даже несколько промежуточных ассистентов
- В статье есть некоторые теоретические обоснования, почему это работает



Достаточно ли dark knowledge из выходов softmax?

ИСП РАН

- Что насчет промежуточных активаций?

FitNets (Romero et al. 2014)

- Хотим, чтобы выходы g-го (*guided*) слоя студента были похожи на выходы h-го (*hint*) слоя учителя
- Выходы разного размера, поэтому добавим слой-регрессор (*сверточный*, он эффективнее) с такой же nonlinearity, как у hint:

$$\mathcal{L}_{HT} = \frac{1}{2}\|u_h(\mathbf{x}, \mathbf{W}_{\text{Hint}}) - r(v_g(\mathbf{x}, \mathbf{W}_{\text{Guided}}), \mathbf{W}_r)\|^2$$

- Имеем обученного учителя. Первый этап — обучение студента с 1-го по g-й слой. Второй — KD как у Хинтона

Algorithm 1 FitNet Stage-Wise Training.

The algorithm receives as input the trained parameters $\mathbf{W_T}$ of a teacher, the randomly initialized parameters $\mathbf{W_S}$ of a FitNet, and two indices h and g corresponding to hint/guided layers, respectively. Let $\mathbf{W_{Hint}}$ be the teacher's parameters up to the hint layer h. Let $\mathbf{W_{Guided}}$ be the FitNet's parameters up to the guided layer g. Let $\mathbf{W_r}$ be the regressor's parameters. The first stage consists in pre-training the student network up to the guided layer, based on the prediction error of the teacher's hint layer (line 4). The second stage is a KD training of the whole network (line 6).

Input: $\mathbf{W_S}, \mathbf{W_T}, g, h$

Output: $\mathbf{W_S^*}$

1: $\mathbf{W_{Hint}} \leftarrow \{\mathbf{W_T}^1, \ldots, \mathbf{W_T}^h\}$

2: $\mathbf{W_{Guided}} \leftarrow \{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\}$

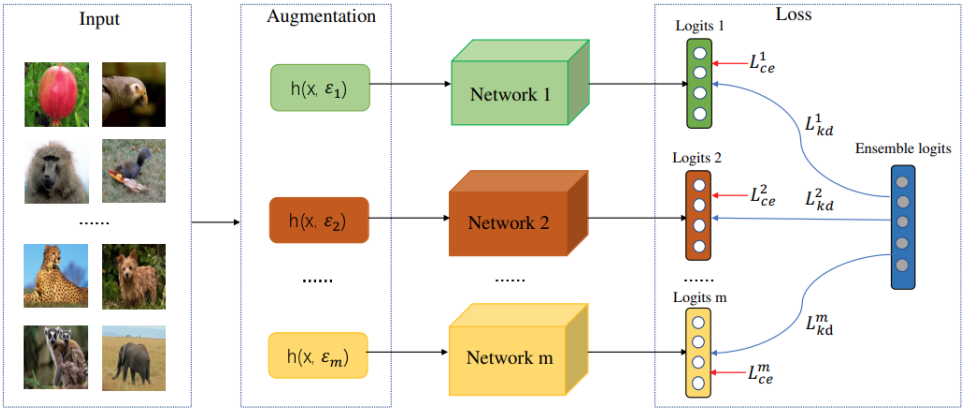
3: Intialize $\mathbf{W_r}$ to small random values

4: $\mathbf{W^*_{Guided}} \leftarrow \underset{\mathbf{W_{Guided}}}{\text{argmin}}\ \mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r})$

5: $\{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\} \leftarrow \{\mathbf{W_{Guided}}^{*1}, \ldots, \mathbf{W_{Guided}}^{*g}\}$

6: $\mathbf{W_S^*} \leftarrow \underset{\mathbf{W_S}}{\text{argmin}}\ \mathcal{L}_{KD}(\mathbf{W_S})$

- Guo et al. 2020



Идея

- Soft label генерируется (и *обновляется*) всеми студентами; предобученный учитель с фиксированными soft label в методе отсутствует
- Пусть у нас m сетей (студентов), имеем их логиты $\mathbf{z}_1, \ldots, \mathbf{z}_m$
- Авторы предлагают несколько вариантов генерации soft label (логитов “коллективного учителя”):

$$\mathbf{z}_t = h(\mathbf{z}_1, \ldots, \mathbf{z}_m)$$

Варианты генерации soft label

ИСП РАН

KDCL-Naive

- Просто берем один из логитов студентов с минимальной кросс-энтропией (сравниваем с реальной меткой y)

KDCL-Linear

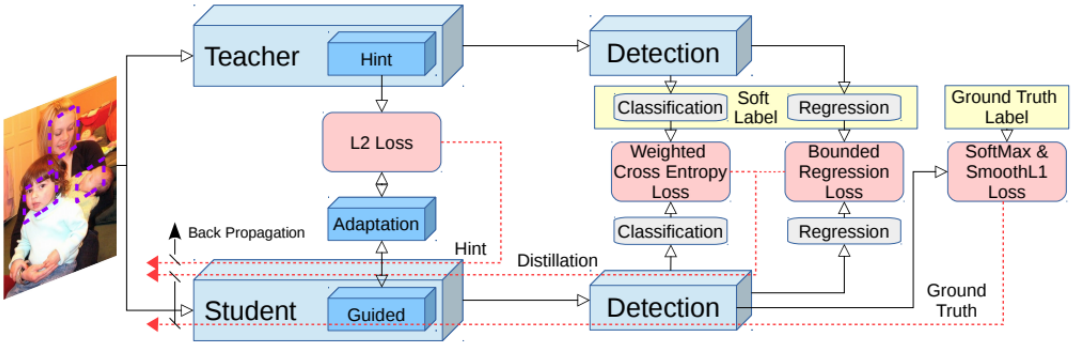
- Ищем хорошую (в смысле кросс-энтропии) линейную комбинацию:

$$\min_{\boldsymbol{\alpha}} L_{CE}(\boldsymbol{\alpha}^T \mathsf{Z}, \mathsf{y}) \qquad \sum_i \alpha_i = 1 \quad \alpha_i \geq 0$$

Дистилляция для object detection (Chen et al. 2017)

ИСП РАН

- Модификация Faster-RCNN



ИСП РАН

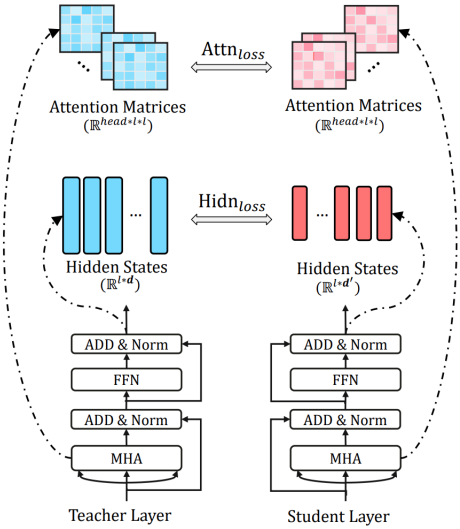
Teacher-bounded regression loss

- Выходы регрессора учителя — произвольные вещественные числа (в отличие от классификации), что может быть опасно
- Поэтому измеряем ℓ_2 с реальным bounding box:

$$L_b(R_s, R_t, y) = \begin{cases} \|R_s - y\|_2^2 & \text{если } \|R_s - y\|_2^2 + m > \|R_t - y\|_2^2 \\ 0 & \text{иначе} \end{cases}$$

- Если студент сильно превосходит учителя, то лосс равен 0

Дистилляция BERT: TinyBERT (Jiao et al. 2019)



ИСП РАН

Процесс

- Есть учитель с N Transformer слоями и необученный студент с $M < N$ Transformer слоями
- Выберем M из N слоев учителя и введем correspondence loss между разными компонентами слоев:

$$\mathcal{L}_{\text{model}} = \sum_{x} \sum_{m=0}^{M+1} \lambda_m \mathcal{L}_{\text{layer}}(f_m^S(x), f_{g(m)}^T(x))$$

Соответствие выходов embedding слоя

$$\mathcal{L}_{\text{embd}} = \text{MSE}(\mathbf{E}^S\mathbf{W}_e, \mathbf{E}^T)$$

Соответствие скрытых представлений

$$\mathcal{L}_{\text{hidn}} = \text{MSE}(\mathbf{H}^S\mathbf{W}_h, \mathbf{H}^T)$$

Соответствие attention матриц

$$\mathcal{L}_{\text{attn}} = \frac{1}{h}\sum_{i=1}^{h} \text{MSE}(\mathbf{A}_i^S, \mathbf{A}_i^T)$$

h — количество attention heads, в $\mathbf{A}_i \in \mathbb{R}^{l\times l}$ l — sequence length

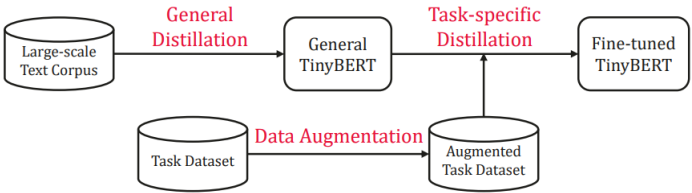
Соответствие предсказаний

- Кросс-энтропия с температурой

Итоговый лосс для каждого слоя

$$\mathcal{L}_{\text{layer}} = \begin{cases} \mathcal{L}_{\text{embd}} & m = 0 \\ \mathcal{L}_{\text{hidn}} + \mathcal{L}_{\text{attn}} & 0 < m \leq M \\ \mathcal{L}_{\text{pred}} & m = M + 1 \end{cases}$$

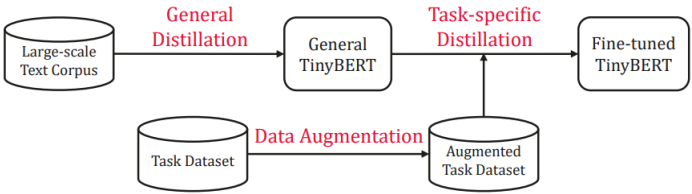
Обучение TinyBERT на downstream задачах



General distillation

- Берем BERT без fine-tuning в качестве учителя, делаем процедуру Transformer distillation
- Из-за сильного уменьшения в размере получается не очень хорошая модель

Обучение TinyBERT на downstream задачах



Task-specific distillation

- Процедура повторяется, но уже с fine-tuned BERT и оригинальной аугментацией (дополнительно использовали GloVe word embeddings)