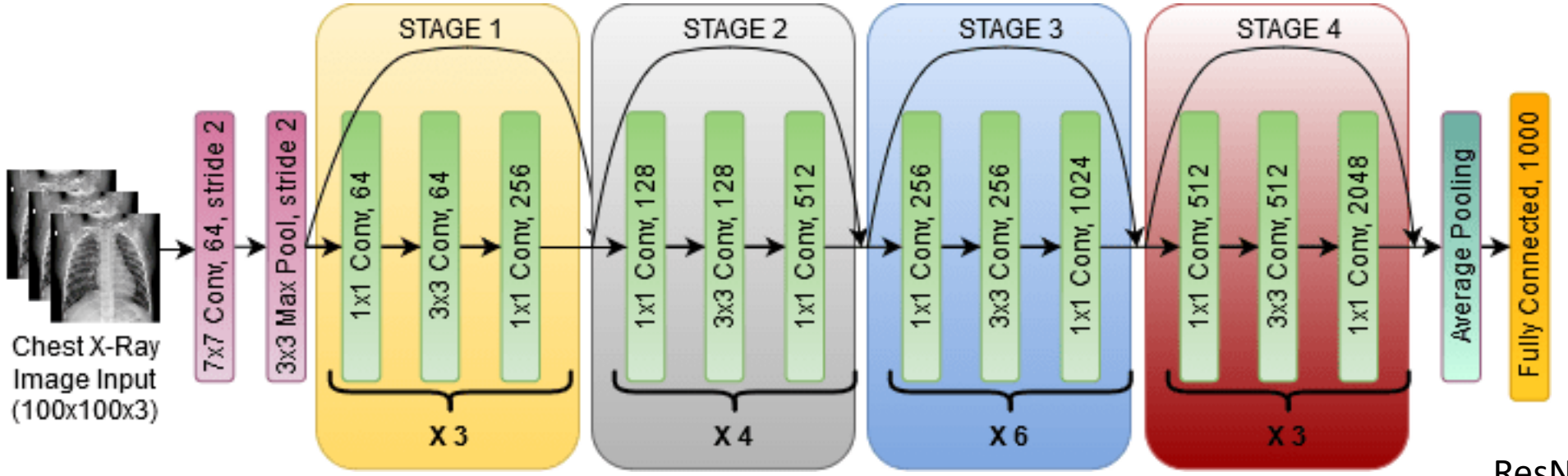
# Лекция 4
# Перенос обучения

Разработка нейросетевых систем

Канев Антон Игоревич

# Предобученные модели




ResNet

model = torch.hub.load("pytorch-cifar-models", "cifar100_mobilenetv2_x0_5", pretrained=**True**)

Задача до

# Предобученная модель распознает виды животных

✓

✗

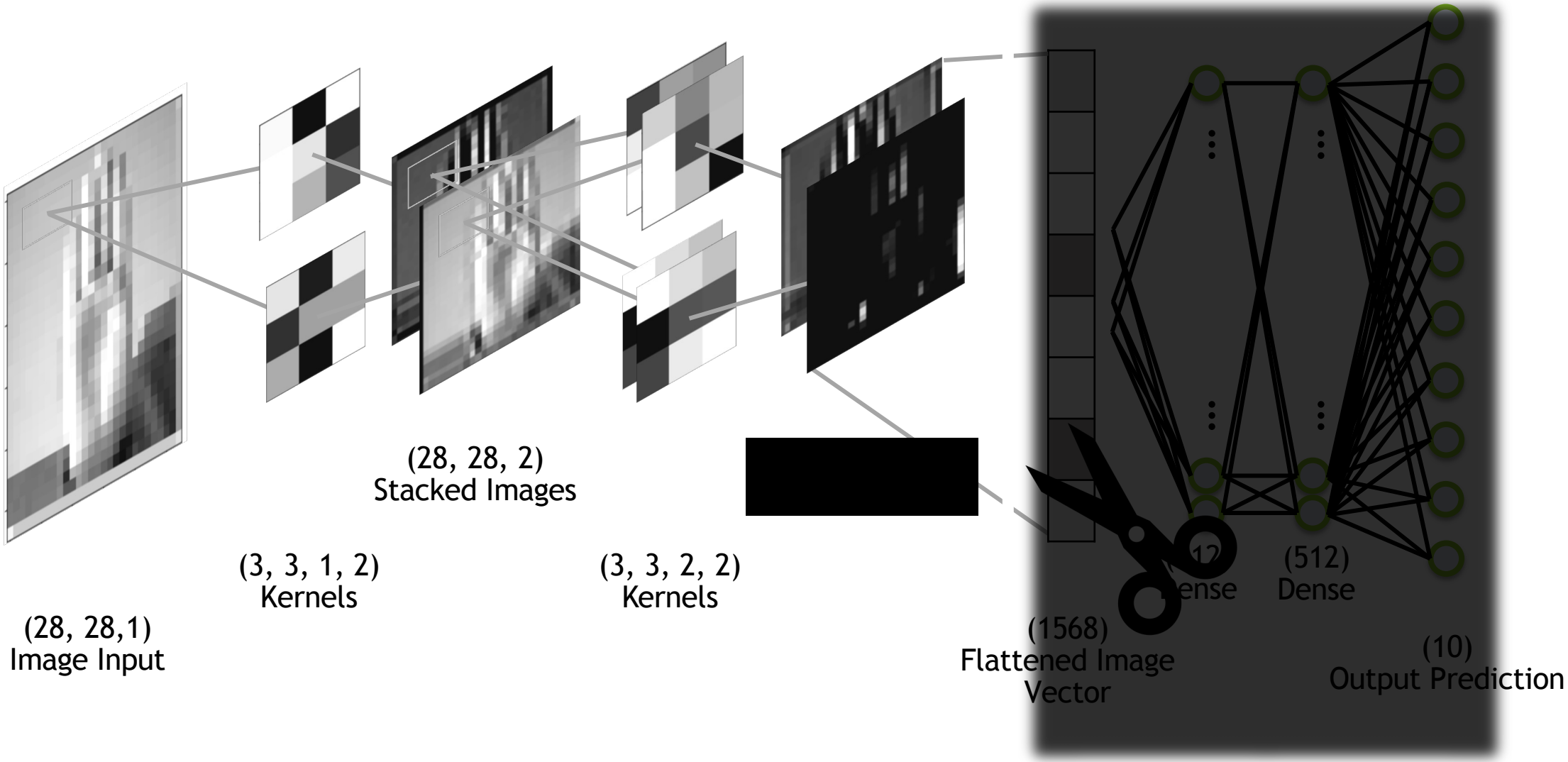
Задача после

# Дообученная модель распознает породы собак

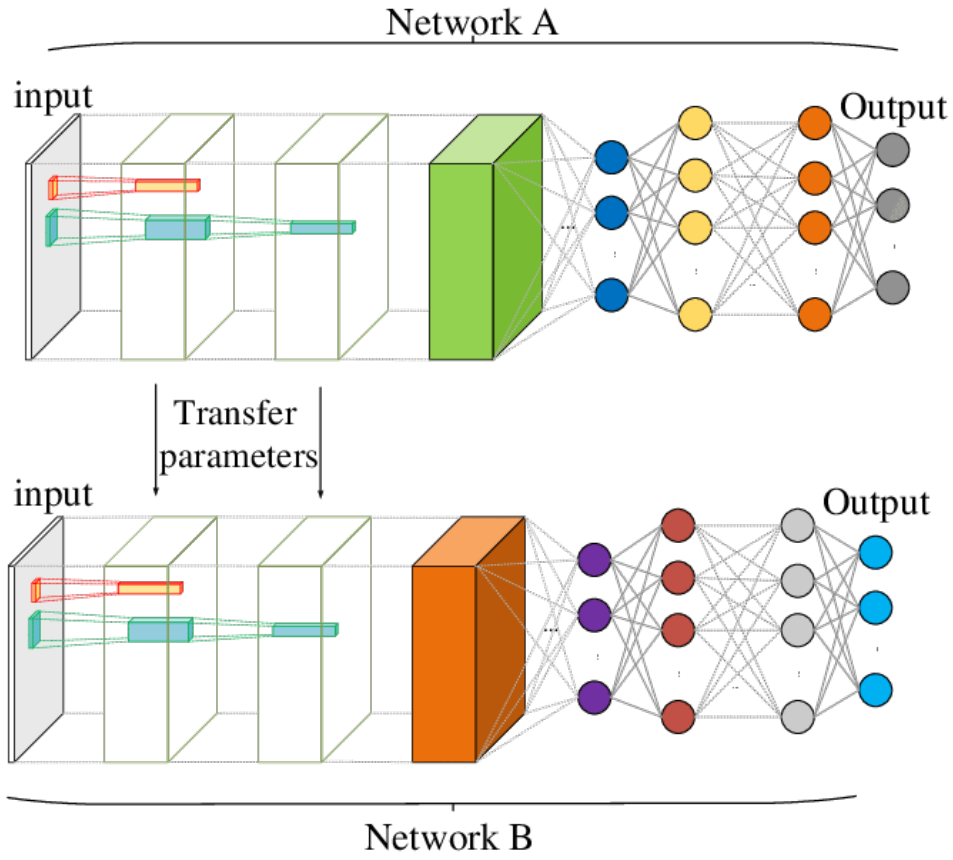
✓

✗

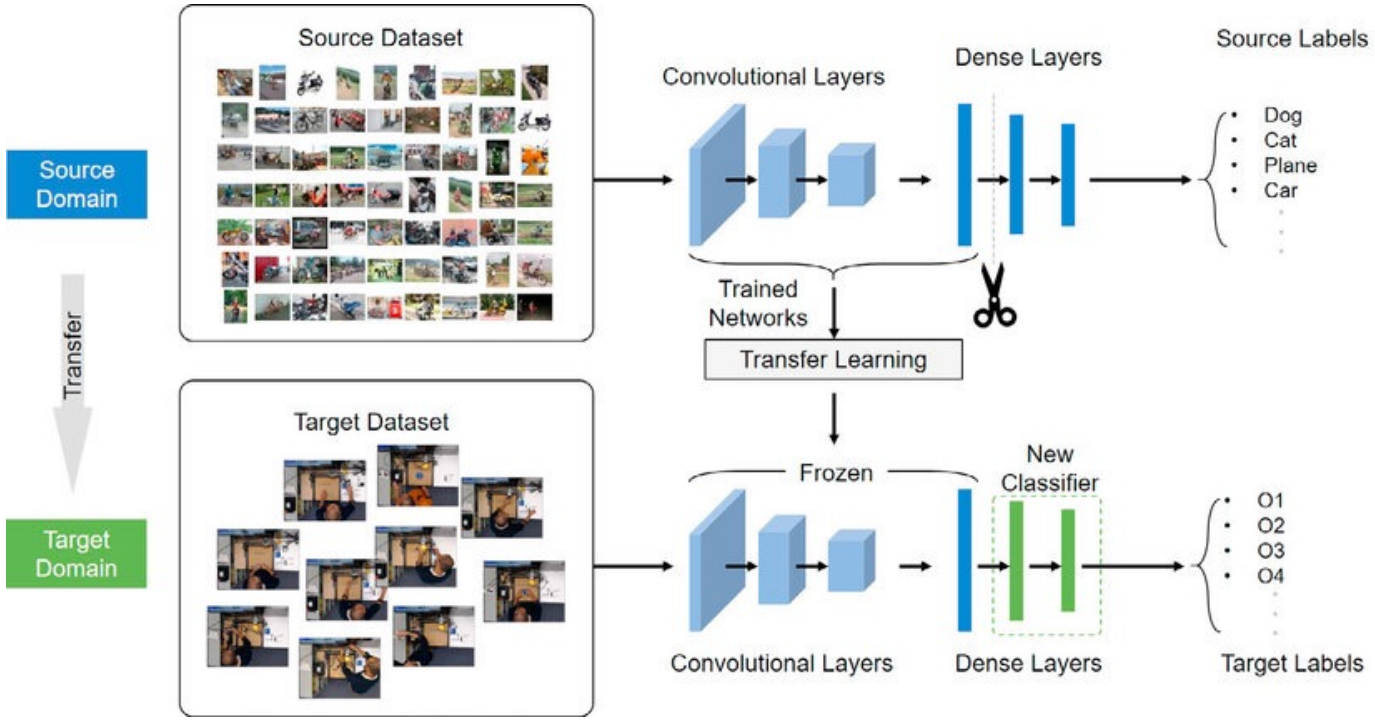
# Перенос обучения



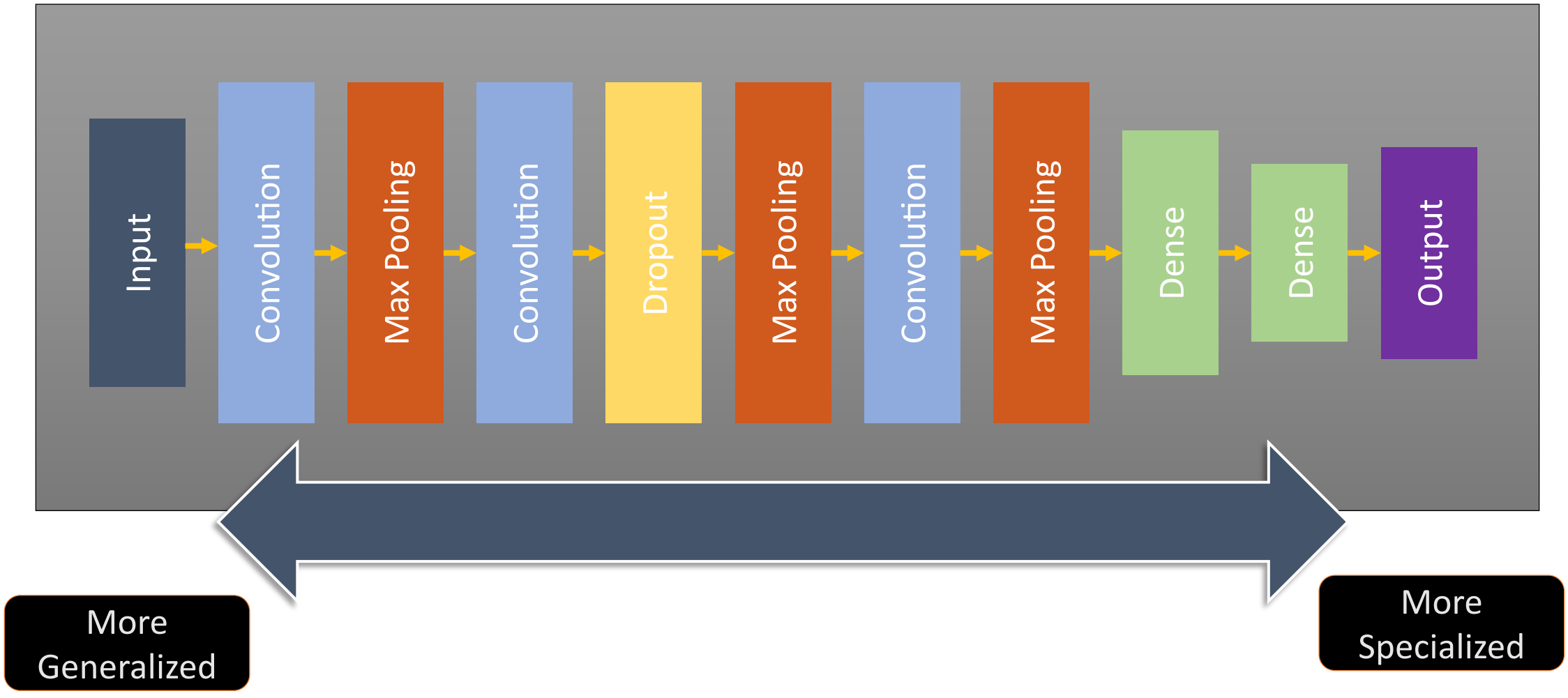
# Перенос обучения  - transfer learning




- Использование готовых моделей
- Для близких задач на новых класса можно применить transfer learning



# Transfer Learning



# Заморозка модели

- После заморозки весов преобученной модели наступает этап обучения новых слоев
- Заморозка нужна чтобы не испортить уже обученные веса

```
for i, (name, param) in enumerate(new_model.named_parameters()):
    if i < total - keep_last:
        param.requires_grad = False
    else:
        params_to_update.append(param)
        print("\t",name)
        param.requires_grad = True
```

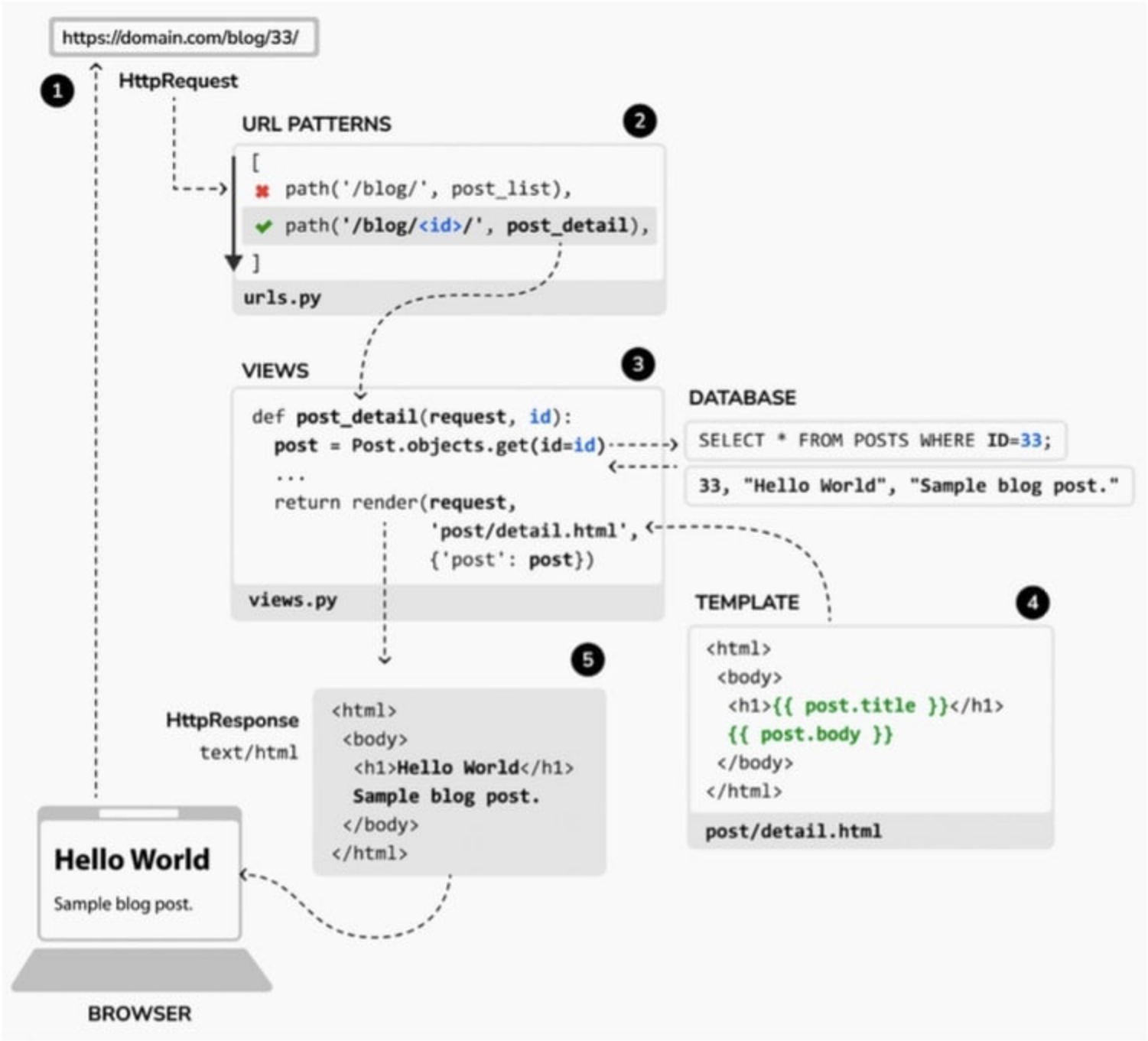
```
Обучаемые параметры:
        1.fc.weight
        1.fc.bias


Total params: 272,019
Trainable params: 195
Non-trainable params: 271,824
```

# Fine Tuning- дообучение

- Когда последние слои не обучены, они вносят большую ошибку и через обратное распространение влияют на первые
- После обучения выходных слоев все остальные веса можно разморозить и продолжить обучение
- Это позволяет тонко настроить все веса под нашу задачу и еще больше повысить точность

# Django

- Django – это MVC фреймворк
- При обработке запроса сначала обрабатывается URL
- Решается, какой view будет его обрабатывать
- View обращается к БД или нейросети
- Результаты вносятся в шаблон Template, получается HTML



# Стохастический градиентный спуск

$L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i)$ – значение функции потерь
$f(\boldsymbol{x}(i); \boldsymbol{\theta})$ – результат вычисления нейронной сети от входа $\boldsymbol{x}(i)$ и параметров (весов) $\boldsymbol{\theta}$

Обновление на k-ой итерации стохастического градиентного спуска (СГС)
**Require**: скорость обучения $\epsilon_k$
**Require**: Начальные значения параметров $\boldsymbol{\theta}$
**while** критерий остановки не выполнен do
Выбрать из обучающего набора мини-батч m примеров $\{\boldsymbol{x}(1), \dots, \boldsymbol{x}(m)\}$ и соответствующие им метки y(i).
Вычислить оценку градиента: $g \leftarrow + (1/\mathrm{m}) \nabla_{\boldsymbol{\theta}} \Sigma_{\mathrm{i}} L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.
Применить обновление:$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon g$.
**end while**

# Adagrad

Алгоритм AdaGrad по отдельности адаптирует скорости обучения всех параметров модели.
Для параметров, по которым частная производная функции потерь наибольшая, скорость обучения уменьшается быстро, а если частная производная мала, то и скорость обучения уменьшается медленнее.
В итоге больший прогресс получается в направлениях пространства параметров со сравнительно пологими склонами

Алгоритм AdaGrad
Require: глобальная скорость обучения $\varepsilon$ Require: начальные значения параметров $\boldsymbol{\theta}$
Require: небольшая константа $\delta$, например $10{-}7$, для обеспечения численной устойчивости.
Инициализировать переменную для агрегирования градиента $\boldsymbol{r}{=}\boldsymbol{0}$
while критерий остановки не выполнен do
Выбрать из обучающего набора мини-пакет m примеров {x(1), ..., x(m)} и
соответствующие им метки y(i).
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/\mathrm{m})\, \nabla_{\boldsymbol{\theta}} \Sigma_i L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.

Агрегировать квадраты градиента: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$.
Вычислить обновление: $\Delta\boldsymbol{\theta} \leftarrow -\epsilon \,/\, (\,\delta + \sqrt{r}\,) \odot \boldsymbol{g}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$. end while

# RMSProp

AdaGrad уменьшает скорость обучения, принимая во внимание всю историю квадрата градиента, и может случиться так, что скорость станет слишком малой еще до достижения такой выпуклой структуры.
В алгоритме RMSProp используется экспоненциально затухающее среднее, т. е. далекое прошлое отбрасывается

**Require**: глобальная скорость обучения $\varepsilon$, скорость затухания $\rho$ Require: начальные значения параметров $\boldsymbol{\theta}$
**Require**: небольшая константа $\delta$, например $10^{-6}$, для стабилизации деления на малые числа
Инициализировать переменную для агрегирования градиента $\boldsymbol{r} = 0$
**while** критерий остановки не выполнен **do**
Выбрать из обучающего набора мини-пакет $m$ примеров $\{\boldsymbol{x}(1), \ldots, \boldsymbol{x}(m)\}$ и соответствующие им метки $\boldsymbol{y}i$.
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/\mathrm{m})\, \nabla_{\boldsymbol{\theta}}\, \Sigma_i\, L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.

Агрегировать квадраты градиента: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho)\, \boldsymbol{g} \odot \boldsymbol{g}$.
Вычислить обновление параметров: $\Delta\boldsymbol{\theta} \leftarrow - \epsilon / \sqrt{(\delta + r)} \odot \boldsymbol{g}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
**end while**

# Импульсный метод

Стохастический градиентный спуск (СГС) с учетом импульса

**Require**: скорость обучения $\varepsilon$, параметр импульса $\alpha$

**Require**: начальные значения параметров $\theta$, начальная скорость $v$

**while** критерий остановки не выполнен do

Выбрать из обучающего набора мини-пакет m примеров $\{x(1), \ldots, x(m)\}$ и соответствующие им метки $y(i)$.

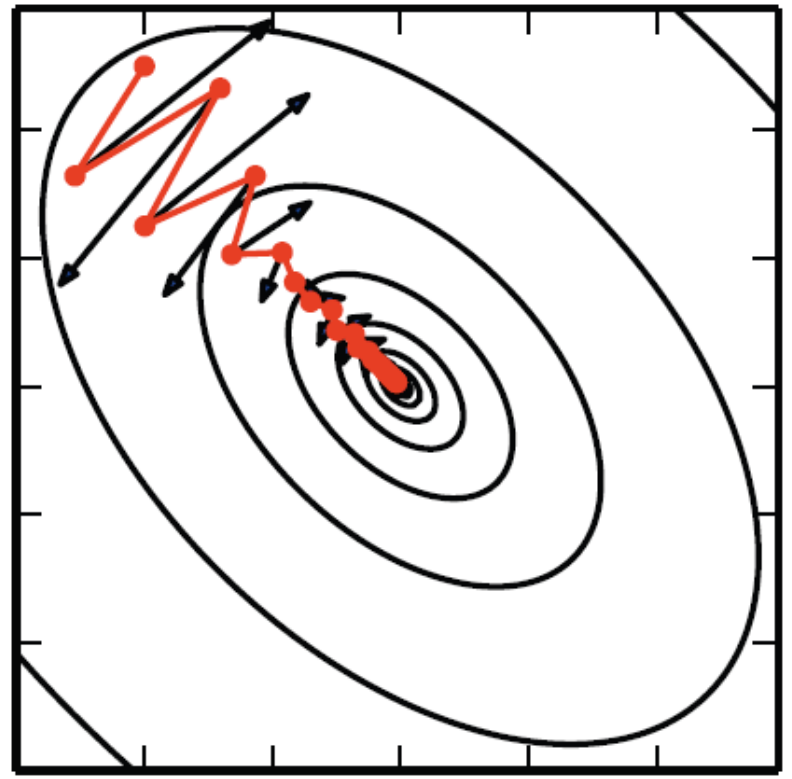
Вычислить оценку градиента:

$g \leftarrow (1/\mathrm{m})\, \nabla_\theta \Sigma_\mathrm{i}\, L(f(x(i); \theta), y(i))$.

Вычислить обновление скорости: $v \leftarrow \alpha v - \varepsilon g$.

Применить обновление: $\theta \leftarrow \theta + v$.

**end while**



Импульсный алгоритм можно рассматривать как имитацию движения частицы, подчиняющейся динамике Ньютона.

# Adam

«Adam» – сокращение от «adaptive moments» (адаптивные моменты). Его правильнее всего рассматривать как комбинацию RMSProp и импульсного метода

**Require**: величина шага $\varepsilon$ (по умолчанию 0.001).
**Require**: коэффициенты экспоненциального затухания для оценок моментов $\rho$ и $\rho$ , принадлежащие диапазону [0, 1) (по умолчанию 0.9 и 0.999 соответственно).
**Require**: небольшая константа $\delta$ для обеспечения численной устойчивости (по умолчанию $10^{-8}$).
**Require**: начальные значения параметров $\boldsymbol{\theta}$.
Инициализировать переменные для первого и второго моментов $\boldsymbol{s} = \mathbf{0}, \boldsymbol{r} = \mathbf{0}$
Инициализировать шаг по времени $t = 0$

**while** критерий остановки не выполнен do
Выбрать из обучающего набора мини-пакет $m$ примеров $\{\boldsymbol{x}(1), \dots, \boldsymbol{x}(m)\}$ и соответствующие им метки $y_i$.
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/m)\nabla\boldsymbol{\theta}\Sigma i\, L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.
$t \leftarrow t+1$
Обновить смещенную оценку первого момента: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Обновить смещенную оценку второго момента: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Скорректировать смещение первого момента: $s \leftarrow s/(1 - \rho_t)$
Скорректировать смещение второго момента: $r \leftarrow r/(1 - \rho_t)$
Вычислить обновление: $\delta\theta = -\epsilon\, s/\sqrt{(\delta+r)}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
**end while**

# Другие оптимизаторы