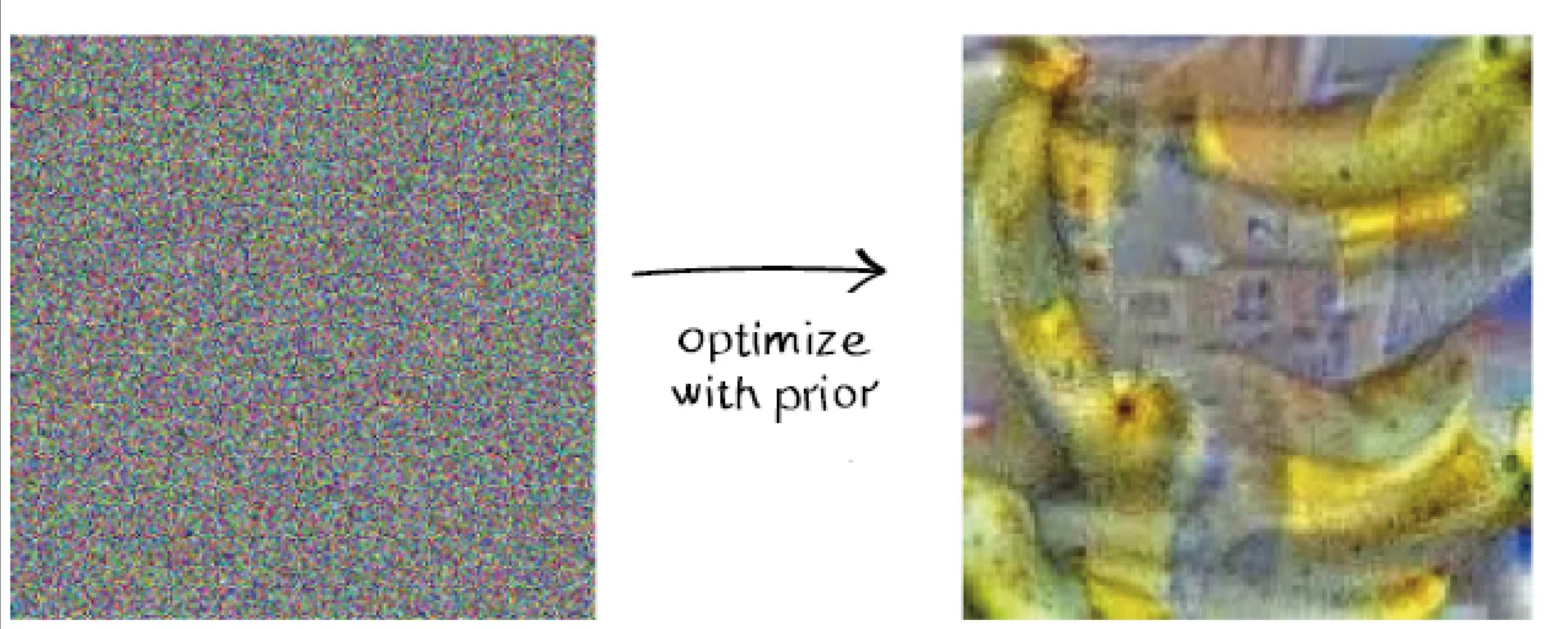




# DeepDream



Один из способов оценить, что именно выучила сеть,  
попробовать восстановить то, что она «видит», взять  
активации с просматриваемых слоёв и утрировать их.



Пример детекции переобучения с помощью *deepdream*,  
видно, как модель, цель которой детектировать гантели  
переобучается и обращает внимание не только на гантели, но  
и на руку

# Алгоритм

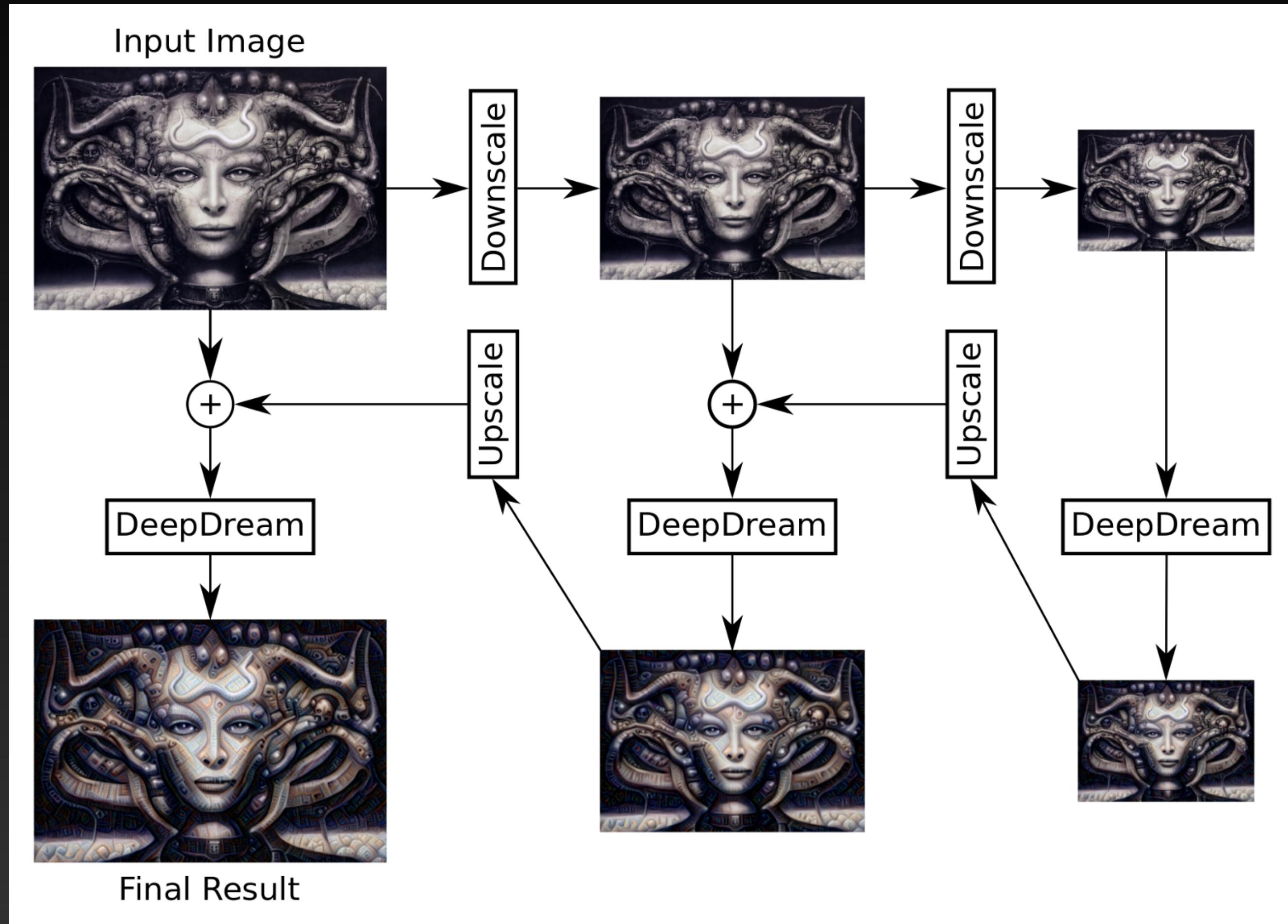
- \* Берет картинку и прогоняет ее через все слои до слоя N
- \* Хотим, чтобы активации на слое N были большие
- \* Считает градиент этого слоя по отношению к входному изображению
- \* Прибавляем этот градиент к исходному изображению



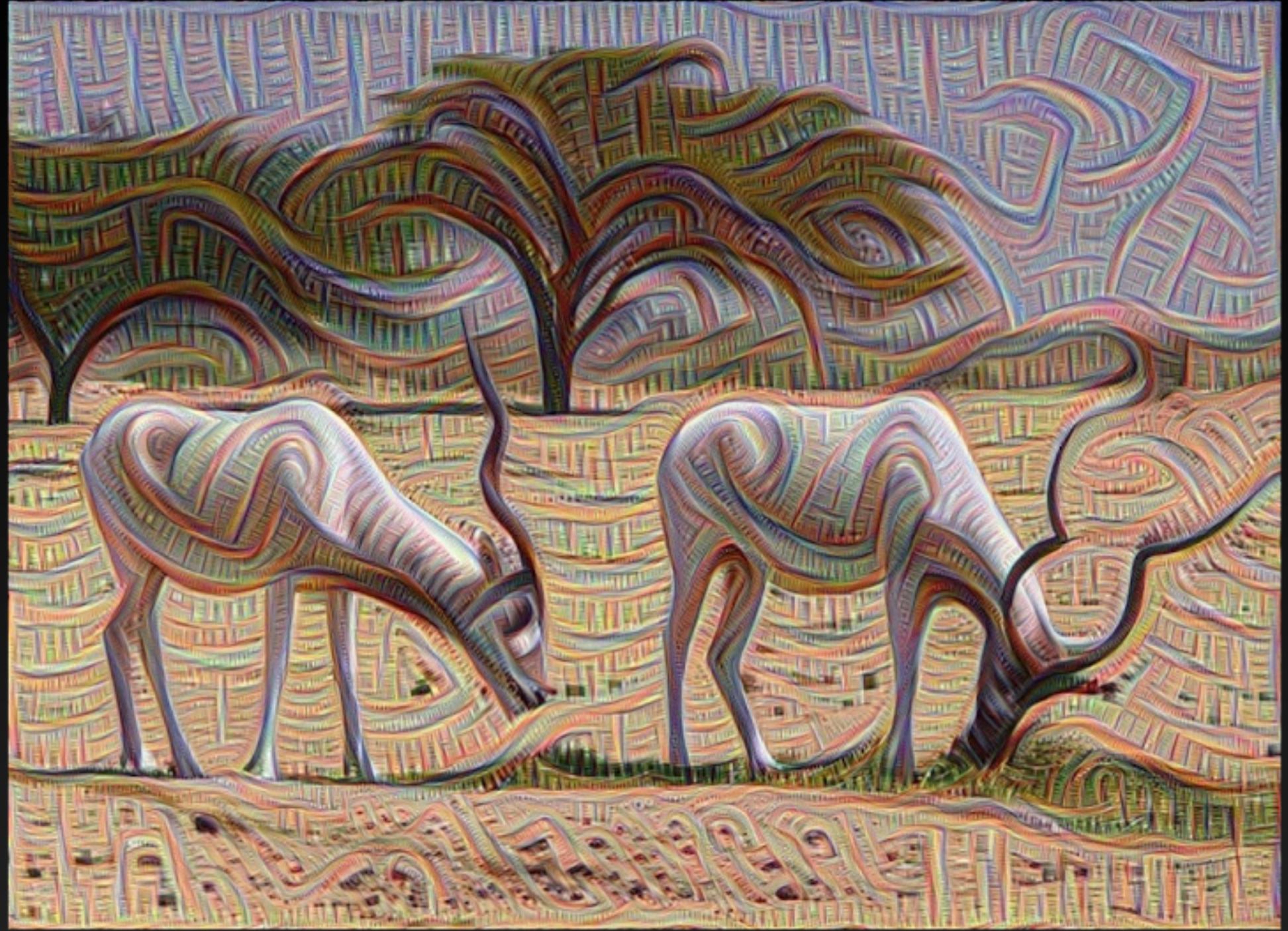
# Но и тут всё не так просто

1. Гауссов блюринг, чтобы сделать изображение «мягче»
2. Использование **«октав»**

# Что за октавы?

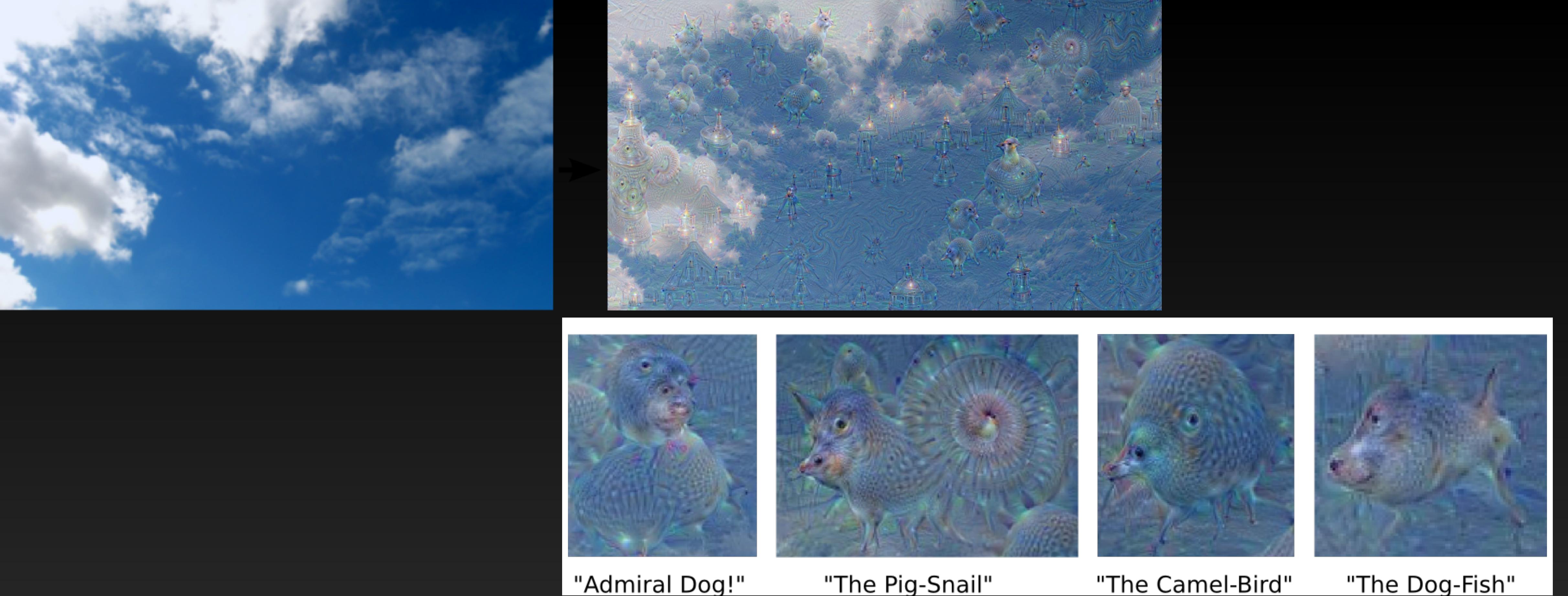


Постоянно уменьшаем размерность изображения, считаем градиент, применяем его к уменьшенному изображению, интерполяционно увеличиваем размер изображения, смешиваем с остальными, созданными на предыдущих этапах



Можем визуализировать,  
например, границы и их  
направление

Для этого визуализируем  
выходы низкоуровневых  
слоёв



В случае визуализации high-level признаков,  
можно увидеть сформированные образы

# Style-transfer





Content Image



Style Image



Generated image

# Классификация алгоритмов NST

1. **Image-Optimisation-Based Online Neural Methods** (IOB-NST). Эта категория передает стиль путем прямого обновления пикселей в изображении итеративно, основана на методах.
2. **Model-Optimisation-Based Offline Neural Methods** (МОВ-NST). Эта категория итеративно оптимизирует генеративную модель и создает стилизованное изображение через один прямой проход, основана на методах.

# IOB-NST



Основная идея алгоритмов IOB-NST состоит в том, чтобы сначала смоделировать и извлечь информацию о стиле и контенте из соответствующих изображений стиля и контента, объединить их в качестве целевого представления, а затем итеративно восстановить стилизованный результат, который соответствует целевому представлению.

# Losses

1. Content loss
2. Style loss

# Content Loss

1. Принимаем на вход фичи из прошлого сверточного слоя для **content\_image** –  $F_{CL}$  – и то же самое для **input\_image** –  $F_{XL}$ . Считаем расстояние между ними и называем это  $D_C^L(X, C) = \|F_{XL} - F_{CL}\|^2$ .
2. Расстояние считаем с помощью ***nn.MSELoss***.
3. Реализуем все это дело как модуль. Само расстояние  $D_C^L(X, C)$  в нем хранится как параметр.

# Style Loss

1. Берем матрицу  $F_{XL}$ , размерностью  $[a, b, c, d]$ , делаем размерность  $[a \cdot b, c \cdot d]$
2. Считаем матрицу Грама  $G_{XL}$  для этой матрицы: это перемножение матрицы на свою транспонированную копию.
3. Нормализуем матрицу Грама: делим каждое ее значение на полное количество элементов в матрице. Это необходимо сделать, чтобы операция получения матрицы Грама не повлияла на повышение активации нейронов.

