|  |  |
| --- | --- |
| **СОГЛАСОВАНО**  Доцент кафедры  ИАНИ ННГУ, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Яшунин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. | **УТВЕРЖДАЮ**  Профессор кафедры  ИАНИ ННГУ, д.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Старостин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |

**Пояснительная записка №1**

**«ПЗ по методам распознавания лиц (ПЗ1)»**

**Этап 1. Подготовка обзоров на существующие подходы к решению   
задачи и построение технологического стека**

**НИР «Разработка и реализация программного обеспечения   
для распознавания лиц на групповых фотографиях»**

**(Шифр ПО «AFR»)**

Ответственный исполнитель

\_\_\_\_\_\_\_\_\_\_\_\_ М.М. Годовицын

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

**Н. Новгород 2019**

**Оглавление**

[1 Введение 2](#_Toc5538641)

[2 Функции потерь 3](#_Toc5538642)

[2.1 Softmax 3](#_Toc5538643)

[2.2 FaceNet (Triplet Loss) 3](#_Toc5538644)

[2.3 Center Loss 4](#_Toc5538645)

[2.4 SphereFace (A-Softmax loss) 5](#_Toc5538646)

[2.5 CosFace(Large margin cosine loss (LMCL)) 6](#_Toc5538647)

[2.6 ArcFace(Additive Angular Margin Loss) 8](#_Toc5538648)

[3 Перенос обучения (Transfer Learning) 9](#_Toc5538649)

[4 Список источников 11](#_Toc5538650)

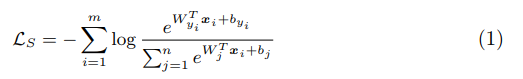
# Введение

В данном документы мы кратко рассмотрим наиболее актуальные способы обучения сети, а конкретно функции потерь (loss functions) использующиеся для вычисления ошибки сети и последующей корректировки весов сети. В следующем разделе рассмотрим способ дообучения сетей или перенос обучения.

# Функции потерь

## Softmax

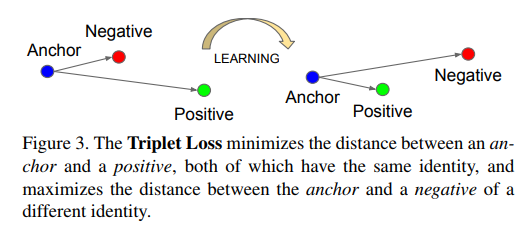
Одна из самых простых функций потерь есть softmax.

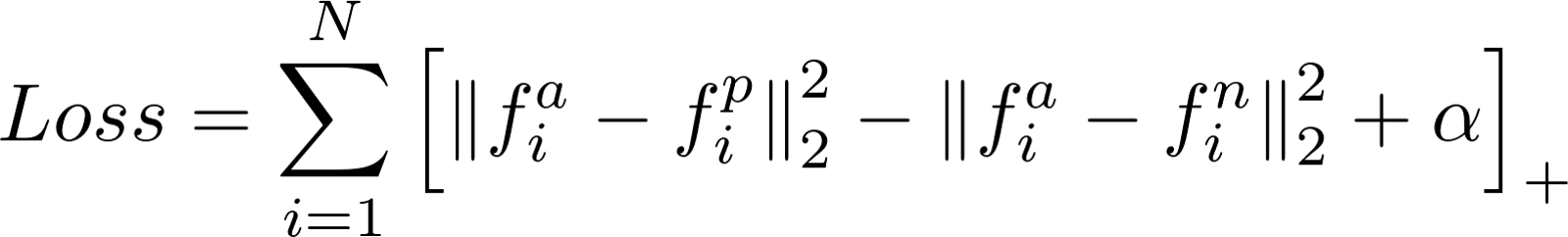


В формуле xi ∈ Rd обозначает i-ую признак, принадлежащую классу yi. d - это размер вектора признаков. Wj ∈ Rd обозначает j-й столбец весов W ∈ R(d × n) в последнем полносвязном слое, а b ∈ Rn - смещение.

## FaceNet (Triplet Loss)

FaceNet использует особую функцию потерь называемую Triplet Loss. Она минимизирует дистанцию между якорем и изображениями, которые содержат ту же личность, и максимизирует дистанцию между различными личностями.





* f(a) это энкодинг якоря
* f(p) это энкодинг похожих лиц (positive)
* f(n) это энкодинг непохожих лиц (negative)
* Альфа — это константа, которая позволяет быть уверенным, что сеть не будет пытаться оптимизировать напрямую f(a) — f(p) = f(a) — f(n) = 0
* […]+ эквивалентно max(0, sum)

## Center Loss

Функция потерь основанная на softmax. Используют линейную классификацию в полносвязном слое. Данная функция потерь учитывает расстояния между объектами и расстояния между объектами и цетром классов, пытаясь балансировать эти оба расстояния. В процессе обучения мы одновременно обновляем центр и минимизируем расстояния между элементами и их соответствующими классовыми центрами. Понятно, что потеря softmax вынуждает разделять признаки разных классов. Потеря центра притягивает к своим центрам признаки того же класса. При совместном обучении увеличиваются не только различия между классами, но и уменьшается внутриклассовое различие. Минус данной функции в том, что обновлять центр для более точного вычисления нужно довольно часто что иногда не совсем просто.

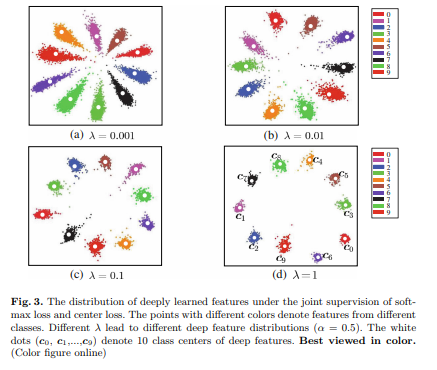


Cyi ∈ Rd обозначает центр классов глубоких признаков. В идеале, cyi должен быть обновлен по мере изменения вектора признаков каждого объекта.

Итоговая функция потерь с гиперпараметром лямбда:

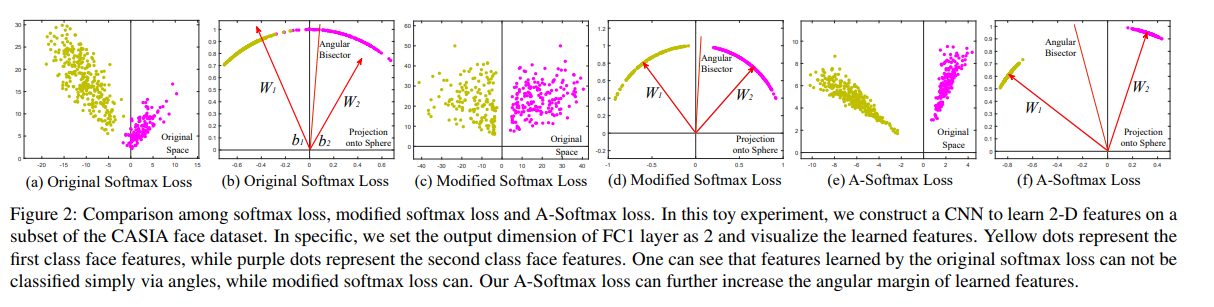
C:\Users\TrueSkit\Desktop\centerloss21.PNG

, где Ls - softmax функция потерь.

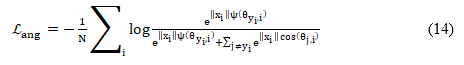


## SphereFace (A-Softmax loss)

Исследователи решили для усиления различительной способности сети добавлять границу вокруг классов объектов, попадание в которую являлось определяющим при принятии решения о классификации объекта.  
Center loss явно поощряет только внутриклассовую компактность.  
Модифицированый softmax (переход в угловое пространство) способна оптимизировать углы, позволяя CNN изучать углово-распределенные элементы.  
Благодаря оптимизации потерь A-Softmax области принятия решений становятся более разделенными, одновременно увеличивая межклассовую границу и уменьшая внутриклассовое угловое распределение.

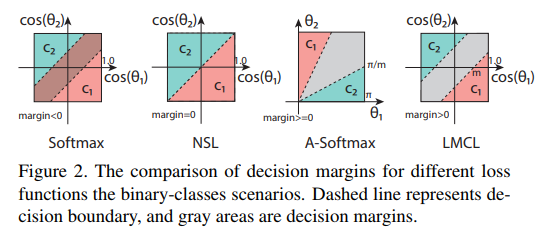


Функция потерь A-Softmax loss:

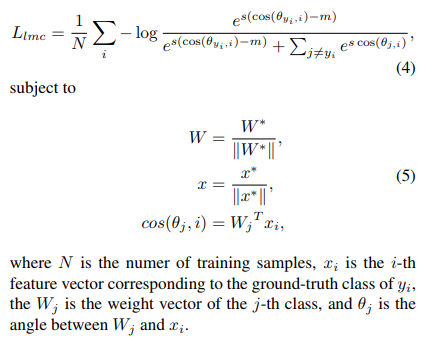
 гдеC:\Users\TrueSkit\Desktop\a-softmax2.png и θyij – угол между векторами признаков i-го примера и шаблона его истинного класса, k – параметр.

# C:\Users\TrueSkit\Desktop\a-softmax3.PNG

## CosFace(Large margin cosine loss (LMCL))

Данная функция потерь берет в себя идеи center loss, large margin softmax loss и angular softmax loss чтобы реализовать всё альтернативным подходом.  Чтобы минимальная внутриклассовая дисперсия и максимальная межклассовая дисперсия достигалась за счет нормализации и максимизации допустимого предела косинус-решения.  
Сначала они переформулируют потерю softmax как потерю косинуса с помощью L2, нормализуя как особенности, так и векторы веса, чтобы устранить радиальные отклонения, на основе которых вводится член m запаса косинуса, чтобы дополнительно максимизировать допустимый запас в угловом пространстве. Формально они определяют гиперпараметр m таким образом, чтобы граница решения определялась как cos (θ1) - m = cos (θ2), где θi - угол между признаком и весом класса i.  


Функция потерь LMCL:



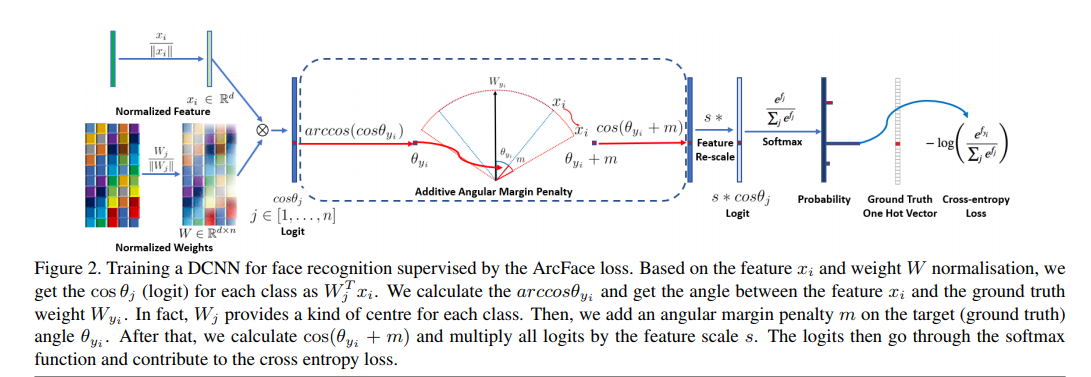
В a-softmax использовалось угловое пространство cos(mθ1) = cos(θ2), что затрудняет оптимизацию из-за немонотонности. Что еще более важно, предел решения A-softmax зависит от θ, что приводит к разным запасам (margin) для разных классов. В результате в области принятия решений некоторые межклассовые признаки имеют больший запас, в то время как другие имеют меньший запас, что уменьшает способность различать.  
В отличие от A-Softmax, их подход определяет границы принятия решения в косинус-пространстве, что позволяет избежать эти недостатки.

## ArcFace(Additive Angular Margin Loss)

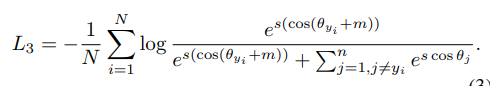
Создателями данной функции потерь предлагается новый способ введения границы между классами, при котором граница принятия решения о принадлежности объекта классу является окружностью фиксированного радиуса на поверхности гиперсферы единичного радиуса в пространстве признаков.  
  
**Sphereface** ввел важную идею углового запаса, их функция потерь требует ряда приближений для вычисления, что привело к нестабильному обучению сети. Чтобы стабилизировать тренировку, они предложили гибридную функцию потери, которая включает стандартную потерю softmax. Эмпирически, потеря softmax доминирует в процессе обучения, потому что мультипликативный угловой запас на основе целочисленных значений делает целевую логит-кривую очень стремительной и, следовательно, препятствует сходимости. **CosFace** напрямую добавляет штраф по косинусному краю к целевому логиту, который обеспечивает лучшую производительность по сравнению со **SphereFace**, но допускает гораздо более простую реализацию и избавляет от необходимости совместного наблюдения за потерями softmax.

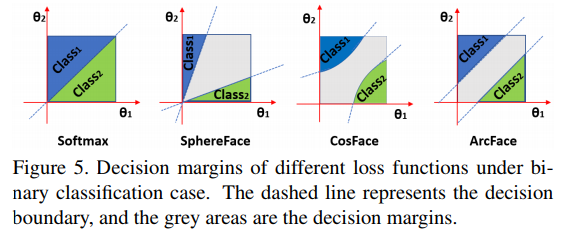
Они предлагают **Additive Angular Margin Loss (ArcFace)**, чтобы дополнительно улучшить дискриминационную силу модели распознавания лиц и стабилизировать процесс обучения.

Как показано на рисунке 2, скалярное произведение между фичей DCNN и последним полносвязным слоем равно расстоянию косинуса после нормализации элемента и веса. Мы используем функцию арккосинус для вычисления угла между текущей фичей и целевым весом. После этого мы добавляем аддитивное угловое поле к целевому углу и снова получаем целевой логит с помощью функции косинуса. Затем мы масштабируем все логиты по фиксированной норме, и последующие шаги точно такие же, как в softmax функции потерь.



Функция потерь Additive Angular Margin Loss:





# Перенос обучения (Transfer Learning)

Технология переноса обучения позволяет использовать готовые нейронные сети для решения задач нового типа, не тех, для которых сети предварительно обучались.  
  
В качестве примера мы рассмотрим, как можно применить предварительно обученные нейронные сети для распознавания кошек и собак.  
  
С помощью технологии переноса обучения мы можем изменить архитектуру предварительно обученной сети таким образом, чтобы она подходила для решения нашей новой задачи. Измененная сеть затем обучается на новом наборе данных.

Нейронные сети, обученные для решения задач классификации изображений, состоят из двух частей:

* **Сверточная часть** используется для выделения характерных признаков из изображения.
* **Полносвязная часть** реализует классификацию - определяет, что за объект находится на изображении на основе признаков, которые извлекла сверточная часть.

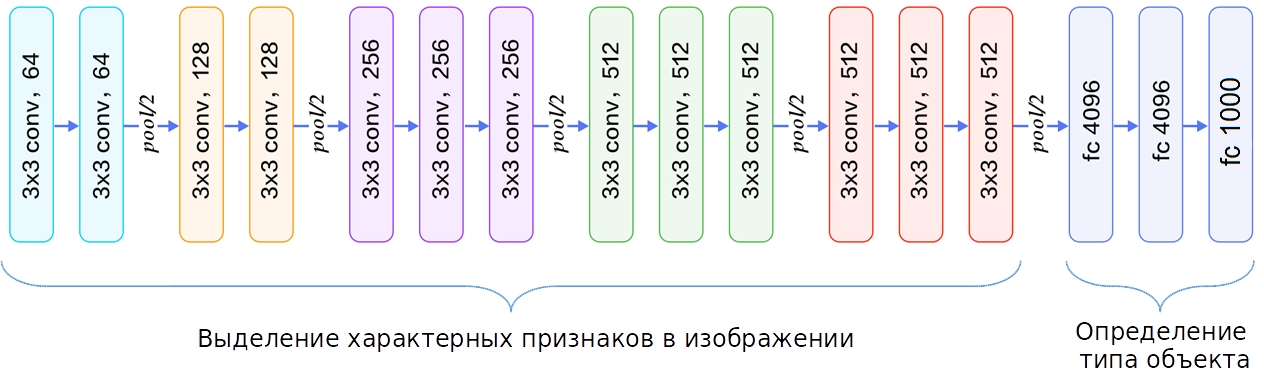
Идея переноса обучения заключается в следующем. Сверточная часть сети во время обучения учится выделять характерные признаки на изображениях. Если признаки получились достаточно общими, то мы можем взять их и применить для решения другой задачи классификации. Таким образом, мы *переносим* обучение сверточной части сети на новую задачу.  
  
Для реализации переноса обучения нам нужно заменить классификатор в предварительно обученной нейронной сети. Давайте рассмотрим как это сделать на примере сети VGG16. Эта сеть достаточно просто устроена и ее легко понять, но в то же время качество работы сети довольно высокое. Архитектура сети VGG16 показана на рисунке 1.  
  


Рисунок Архитектура VGG16  
  
Сверточная часть сети VGG16 состоит из пяти каскадов свертки и подвыборки. В первых двух каскадах используются по два слоя свертки и слой подвыборки с выбором максимального значения (max pooling). На трех следующих каскадах по три слоя свертки и один слой подвыборки. Размер ядер во всех слоях свертки 3х3.

Полносвязная часть сети VGG16 включает три уровня. На выходном уровне 1000 нейронов по количеству классов объектов. Используется формат one-hot encoding: значение только одного выходного нейрона должно быть близко к единице, остальные близки к нулю. Класс объекта на картинке соответствует нейрону, значение которого близко к единице. Перед выходным слоем в сети VGG16 еще два полносвязных слоя по 4096 нейронов.

На первом этапе необходимо убрать полносвязную часть из сети VGG16. Получится сеть как на рисунке 2.

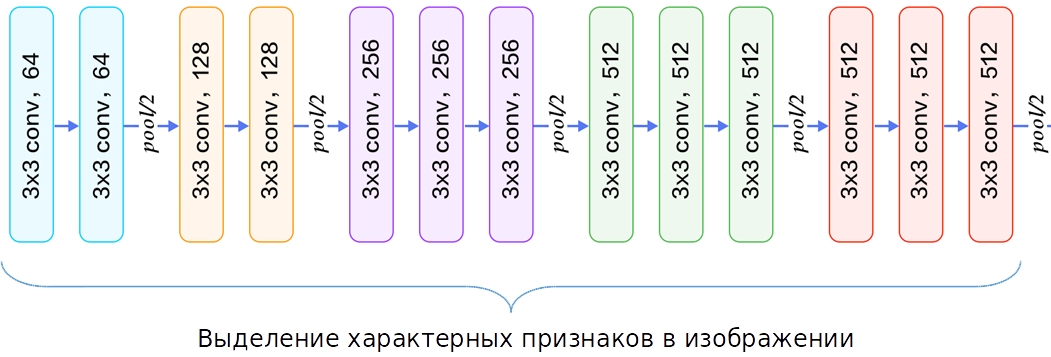


Рисунок 2 Архитектура VGG16 без полносвязного слоя

Второй этап: к сверточной части сети VGG16 добавляем новый классификатор (Рисунок 3).

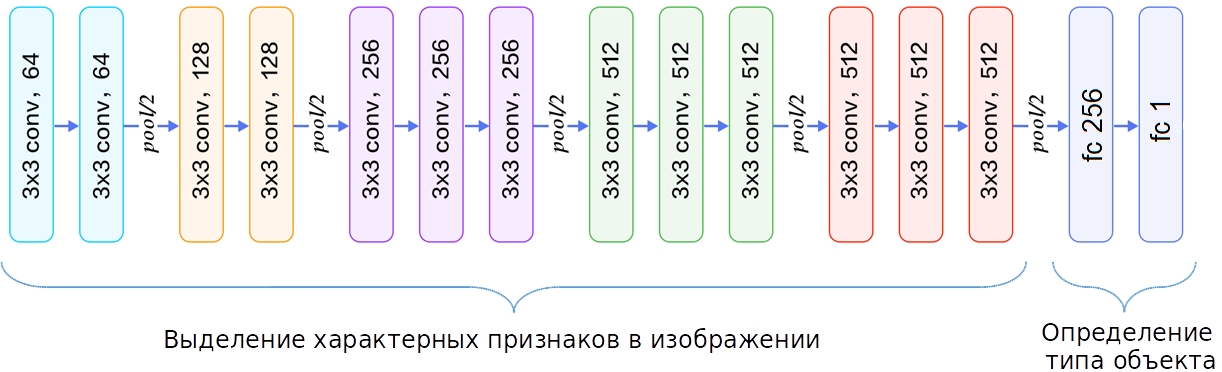


Рисунок 3 Архитектура VGG16 c новым классификатором (кошка или собака)

Новый классификатор устроен гораздо проще полносвязной части сети VGG16, т.к. нам нужно распознавать всего два класса объектов, а не 1000. На выходном слое один нейрон, что соответствует задаче бинарной классификации. Ноль на выходе из сети означает, что на фотографии кот, а единица - собака. Перед выходным слоем находится еще один полносвязный слой, в котором 256 нейронов. На вход этого слоя поступают данные из сверточной части сети VGG16.

На третьем этапе измененную сеть нужно обучить на новом наборе данных с фотографиями котов и собак.

В процессе обучения новой сети нужно запретить обучать сверточную часть сети, в противном случае веса в сети могут испортиться в процессе обучения с новым классификатором. В классификаторе, который мы добавим к сети, веса нейронов будут инициализированы случайными числами. Поэтому на первых этапах обучения значения ошибки на выходе из сети будет очень большим. По алгоритму обратного распространения ошибки сигнал об ошибке будет передаваться и в сверточную часть сети VGG16, из-за чего веса в ней могут испортиться.

# Список источников

1. <https://arxiv.org/abs/1503.03832>
2. <https://www.researchgate.net/publication/308190438_A_Discriminative_Feature_Learning_Approach_for_Deep_Face_Recognition>
3. <https://arxiv.org/abs/1704.08063>
4. <https://arxiv.org/abs/1801.09414>
5. <https://www.researchgate.net/publication/322674945_ArcFace_Additive_Angular_Margin_Loss_for_Deep_Face_Recognition>
6. <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>
7. <https://www.asozykin.ru/deep_learning/2018/01/08/Transfer-Learning-in-Keras.html>