1) Как решается проблема, которая заключается в том, что на изображении объект может быть в разном масштабе?

Можно заранее обрабатывать изображения или же использовать СНС. Сверточные нейронные сети обеспечивают частичную устойчивость к изменениям масштаба, смещениям и прочим искажениям. СНС объединяют три архитектурных идеи, для обеспечения инвариантности к изменению масштаба, повороту, сдвигу и пространственным искажениям:

- локальные рецепторные поля (обеспечивают локальную двумерную связность нейронов);
- общие синаптические коэффициенты (обеспечивают детектирование некоторых черт в любом месте изображения и уменьшают общее число весовых коэффициентов);
- иерархическая организация с пространственными подвыборками.

2) Какой вид обучения используется для решения задачи кластеризации?

Обучение без учителя, так как известны только описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, закономерности, существующие между объектами. В данной задаче нам неизвестно к какому классу относится образец, поэтому мы используем обучение без учителя.

3) Для чего может быть использована свертка с ядром 1x1?

Для изменения размерности пространства фильтра, уменьшать или увеличивать глубину тензора.

4) В чем разница между методами RMSProp и Adagrad?

Математически, функция нейронной сети $f_i(x)$ определяется как композиция функций $g_i(x)$, которые в свою очередь представляют композицию других функций. Широко распространенный тип композиции — нелинейная взвешенная сумма $f(x) = \sigma(\sum_i w_i g_i(x))$, где g_i— активации

нейронов с предыдущего слоя, w_i — веса линейной трансформации, а σ — некоторая заданная нелинейная функция активации.

Adagrad (метод адаптивного градиента) эффективно перемасштабирует шаг обучения μ для каждого параметра НС в отдельности, учитывая историю всех прошлых градиентов для этого параметра (идея масштабирования). Это делается путем деления каждого элемента в градиенте ∇f_i, где f_i – функция подсчитанная на i-той части данных, на квадратный корень суммы квадратов прошлых соответствующих элементов градиента.

Перемасштабирование таким способом эффективно уменьшает шаг обучения для параметров, которые имеют большую величину градиента.

Также метод уменьшает сам шаг обучения со временем, так как сумма квадратов увеличивается с каждой итерацией.

При инициализации масштабирующего параметра $g = 0$ формула для пересчета имеет вид

$$g_{t+1} = g_t + \nabla f_i(\theta_t)^2$$

$$\theta_{t+1} = \theta_t + \frac{\mu \nabla f_i(\theta_t)}{\sqrt{g_{t+1} + \epsilon}}$$

где деление выполняется поэлементно, а ϵ — это небольшая константа, введенная для численной стабильности.

RMSprop (метод адаптивного скользящего среднего градиентов) очень похож по принципу работы на метод Adagrad.

Единственное его отличие в том, что шкалирующий член g_t вычисляется, как экспоненциальное скользящее среднее вместо кумулятивной суммы. Это делает g_t оценкой второго момента градиента ∇f и устраняет тот факт, что шаг обучения со временем уменьшается. Правило пересчета:

$$g_{t+1} = \gamma g_t + (1 - \gamma) \nabla f_i(\theta_t)^2$$

$$\theta_{t+1} = \theta_t + \frac{\mu \nabla f_i(\theta_t)}{\sqrt{g_{t+1} + \epsilon}}$$

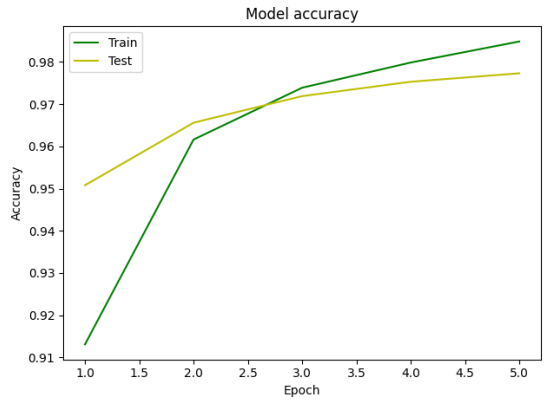
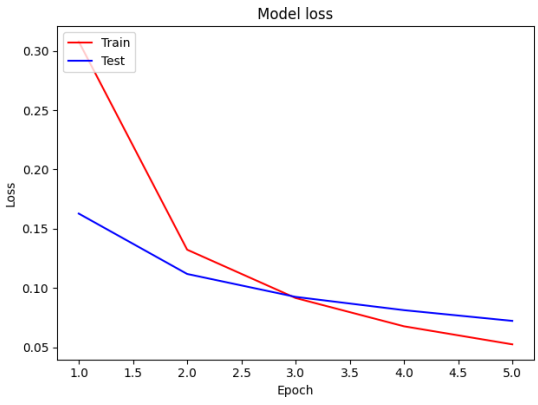
5) Для чего нужен код в строчке 43?

Я открываю изображение в режиме L, это означает, что это одноканальное изображение. Этот режим очень компактный, но хранит только оттенки серого, а не цвета. Получается, я считала изображение, где черный фон и белая цифра. После уменьшения размера изображения, мне нужно перевести его в негатив, так как набор тренировочных данных MNIST имеет белый фон и черную цифру. Для этого я использую код в строчке 43.

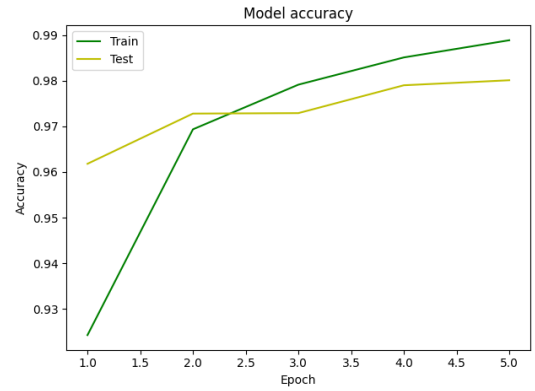
```
img = 1 - img
```

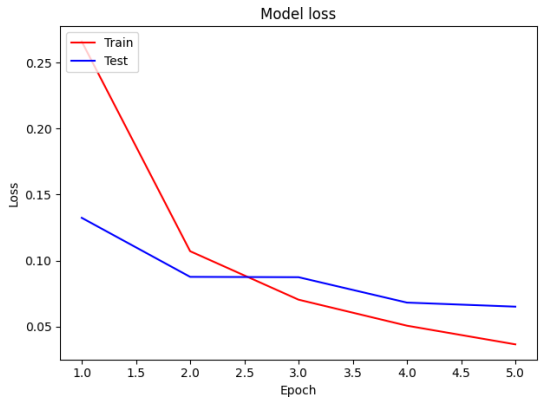
6) Почему выбрана именно такая архитектура сети?

После проведения нескольких тестов со следующими параметрами, я заметила, что с добавлением слоев точность не сильно меняется, поэтому я выбрала архитектуру, при которой наименьшие потери, ориентируясь по графикам. Увеличение и уменьшение нейронов я производила по степеням двойки. Рассмотрим графики:

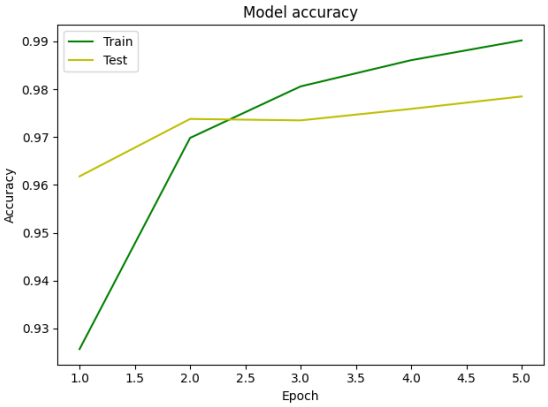
256	
	

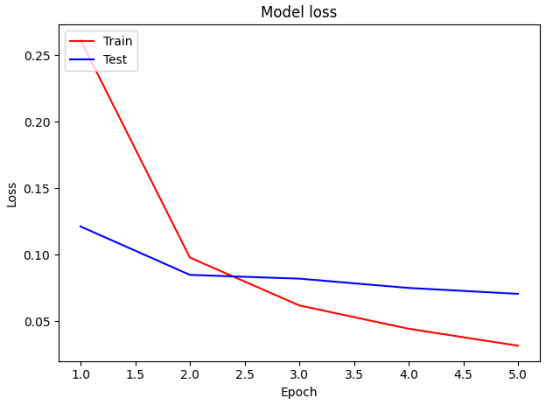
512



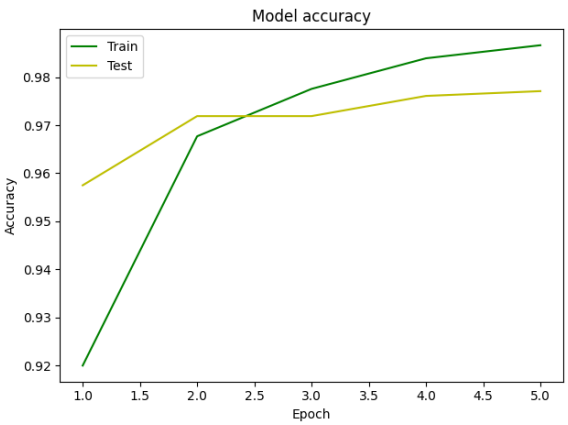


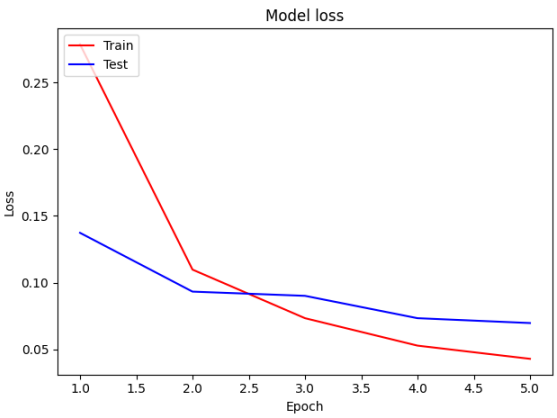
256-256



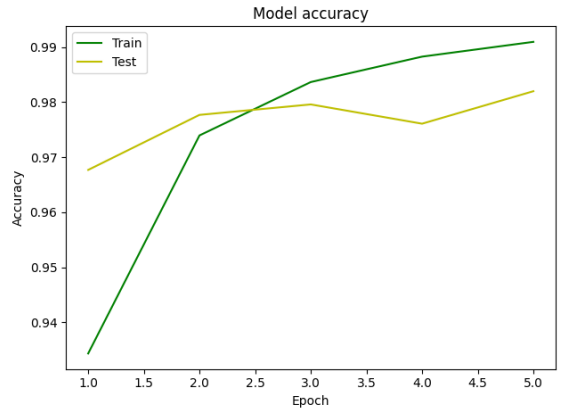


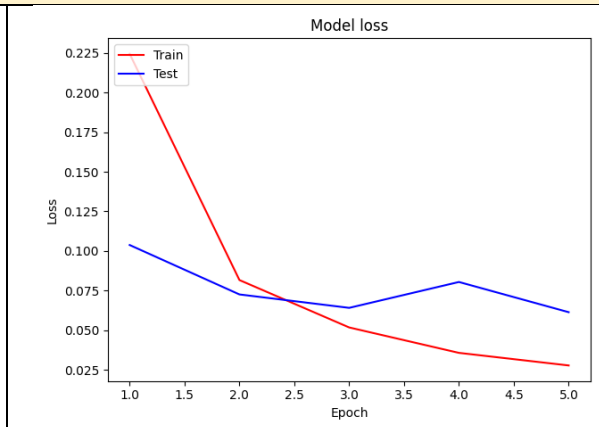
256-64



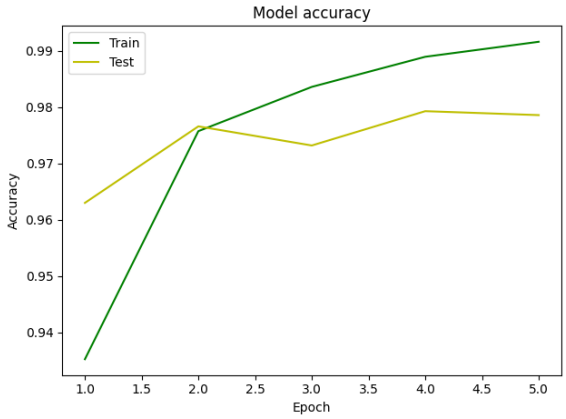


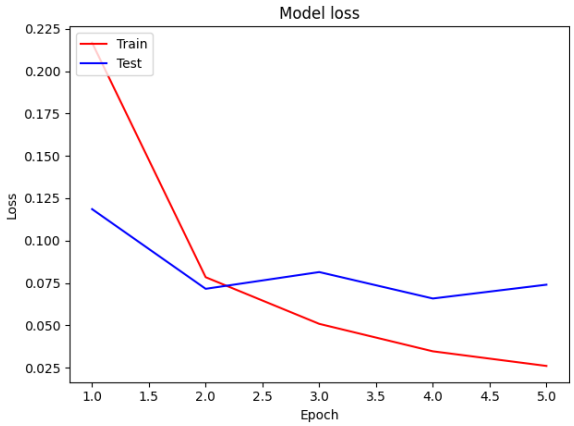
512-256





512-512





Как видно на графиках, точность у архитектуры (512-256) начинает возрастать от 97 %, в остальных же случаях этот показатель меньше. Потери начинают уменьшаться, начиная с 10%, что является наименьшим результатом.