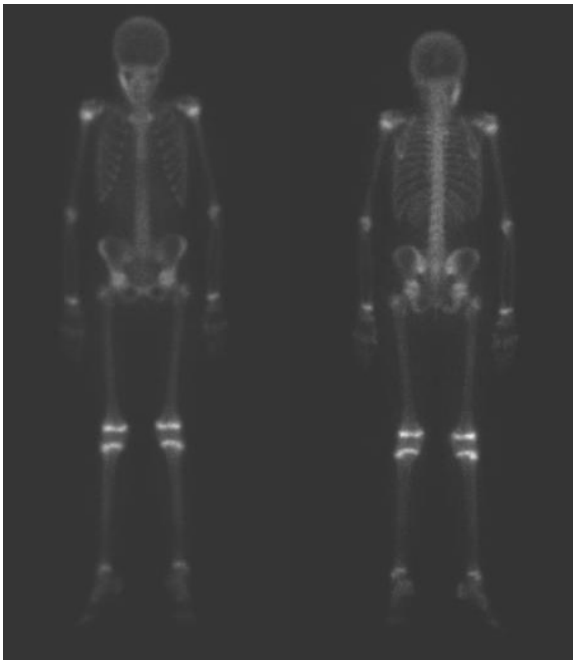
Planáris csontszcintigráfia (csontscan) felvételen végrehajtott zajszűrés

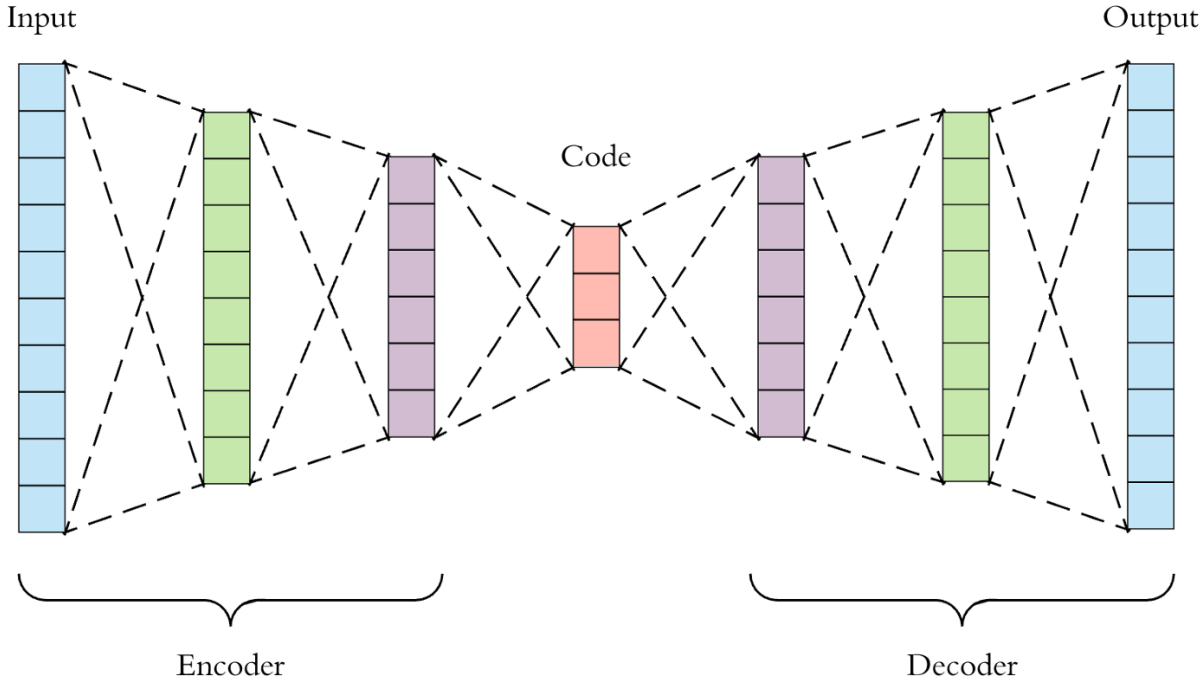
A planáris csontszcintigráfia a leggyakrabban alkalmazott vizsgálat a csont elsődleges és másodlagos (áttétes) daganatainak kimutatásában, követésében és a kezelés eredményességének lemérésében. Maga a felvétel úgy készül, hogy a betegnek vénásan technécium-99m izotóppal jelölt anyagot adnak be, ami a csontokban kerül felhasználásra. A felhasználás során gamma foton hagyja el a testet, amit gamma-kamerával rögzítenek. Így készül egy anterior és poszterior felvétel.



A csontokban található elváltozások nagyobb mennyiségben veszik fel a beadott anyagot, emiatt a felvételen jobban világítanak, ezen kívül a beadási ponton, a húgyhólyagban és enyhén a vesékben is több anyag gyülemlik fel. A képek zajossága nagyban megnehezíti a problémás területek megtalálását.

A három féléves projektmunka feladata a planáris felvételeken jól működő zajszűrő módszer kidolgozása. Ennek első lépéseként az előző félévben elemi eszközökkel végeztünk zajszűrést, főként a python OpenCV programcsomagjában található beépített szűrőkkel. Ezek eredményességét, egy zajszűrő segítségével mértük, amely Chen, Zhu és Ann Heng: An efficient statistical method for image noise level estimation cikkében leírtak alapján működött.

Ebben a félévben a célunk az volt, hogy a zajszűrést mélytanulás segítségével valósítsuk meg. Ehhez egy autoencodert használtunk. Az autoencoder egy neurális háló, amelynek alapelve, hogy megtanulja alacsonyabb dimenzióban reprezentálni a bemenetet, ezáltal megragadni annak lényegi részeit, majd ez alapján visszaadni, az elvárt kimenetet.



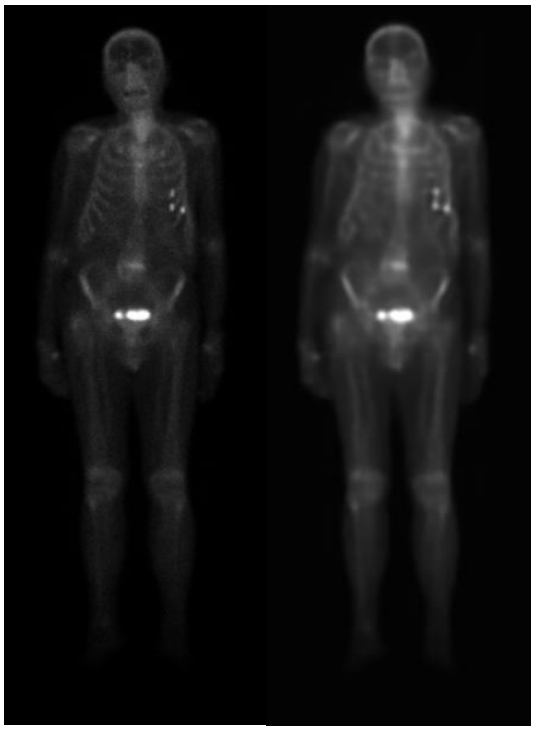
Az autoencoderek három részből állnak. Az első az encoder. Ez a része a hálónak, megkapja bemenetet és előállítja belőle, az ő tömörített és jóval kisebb méretű reprezentációját. Ez sok esetben konvolúciós rétegek és pooling rétegek felhasználásával történik.

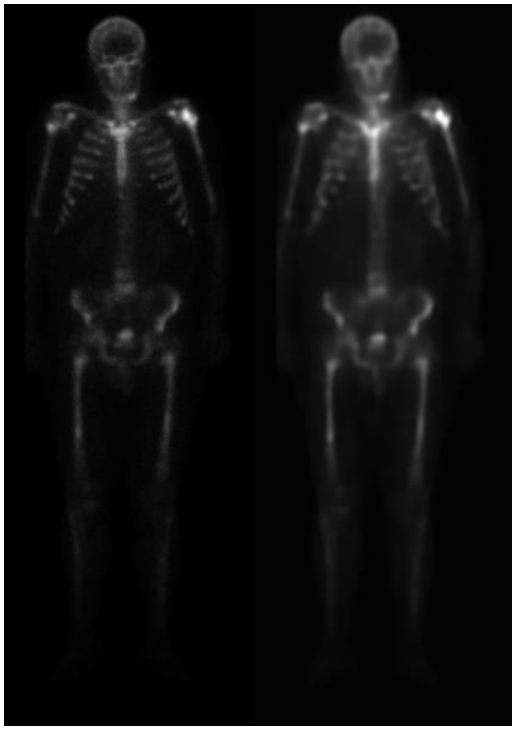
A második fő egysége a code, amely összeköti az encodert a decoderrel. Célja, hogy csak a legfontosabb információkat tárolja az inputból, de azokból lehetőleg mindet. Minél kisebb a mérete, annál kisebb a kockázat, hogy túltanul a háló, viszont, ha túl kicsire állítjuk a méretét akkor értékes információt veszíthetünk.

Végül a decoder segít a reprezentációból felépíteni és rekonstruálni az adatot, amit végül összehasonlítunk az általunk elvárt kimenettel. Így próbáljuk a lehető legtöbb hasznos információt kinyerni. Upsampling és konvolúciós rétegek alkotják. A tanítás lényege, hogy a háló megtanulja az input lényeges tulajdonságait a code előállítása során, majd a megfelelő módon állítsa elő a kimenetet.

Esetünkben a zajszűrés volt a cél, így bemenetnek az általunk vizsgált planáris felvételek mesterségesen zajosított változatát adtuk, és a háló kimenetét az eredeti képpel hasonlítottuk össze. Kétszáz (száz anterior és száz posterior) generált kép állt rendelkezésünkre, mivel ez elég kevés a tanításhoz, ezért az adathalmaz méretének növeléséhez augmentációs eszközöket használtunk. Az előfeldolgozás részeként a 255-nál nagyobb értékeket levágtuk, így minden pixel 0 és 255 közé esett. Első körben tükröztünk mind az x mind az y tengelyre, majd az így kapott hatszáz kép mindegyikéhez generáltunk öt darab véletlenszerűen nagyított/kicsinyített és enyhén forgatott képet. Ezeket betöltöttük egy pytorch Datasetbe, amely során elvégeztük a zajosítást is, minden képhez poisson zajt adtunk a numpy beépített függvényével.

A planáris képek mérete 1024 x 256 pixel. A háló encoder részében konvolúciós (nn.Conv2d) és batch normalizációs (nn.BatchNorm2d) rétegek követték egymást felváltva. Pooling helyett a konvolúciós rétegek stride paraméterét 2-re állítottuk, így értük el a méret csökkenését, a kernel mérete 4 volt a padding 1, a channelek számát mindig dupláztuk. Így az 1 x 1024 x 256 - os inputtól öt lépésben eljutottunk a 256 x 32 x 8 - as code-hoz. A decoder ugyanezen az elven építette fel a kimeneti képet csak itt nn.Co n vTranspose2d-t használtunk. Aktivációs függvénynek relut használtunk. A háló összesen 2576545 paraméterrel rendelkezett. A tanítás 30 epochon keresztül zajlott, 0,001-es learning rate-tel és 50-es batch mérettel, Adam optimizerrel és átlagos négyzetes hibát alkalmaztunk.





Miután betanult a háló, alkalmaztuk 50 valódi anterior felvételre, ezekből látható két pár az előző oldal alján. A bal oldali az eredeti és a jobb oldali az autencoder által készített kép. Jól látható, hogy a szűrt képek mindkét esetben világosabbak, ez mind az ötven képre igaz, valószínűleg azért, mert a háló megtanulta, hogy amikor a tanítás során mesterségesen adtunk zajt a képekhez, akkor jelentősen megnöveltük a pixelek értékeit, így amikor a decoder felépítette a kimenetet az inputéhoz képest jóval nagyobb értékeket adott a pixeleknek.

A félév során jelentős szerepet játszott az új tudás elsajátítása, ezért sajnos nem maradt sok idő az autoencoder tökéletesítésére. Ezért már most sok pontot látunk, ahol fejleszteni, illetve javítani tudjuk a hálót. Az előbb említett világosítás problémájára megoldást jelenthet, ha más módon zajosítjuk a felvételeket, például binomiális szétbontással. A transzponált konvolúciós réteg használata során gyakran okoz képhibákat, ezért érdemes lehet a decoderben lecserélni egy UpSample és egy konvolúciós rétegre. Tervezzük még kipróbálni a problémára a u-net-et, ami egy nagyon hasonló architektúra, és eredményesen alkalmazták egyéb orvosi területeken. Természetes ötlet, hogy megpróbáljuk a háló méretét növelni, amíg az eszközeink engedik, illetve érdemes lehet kipróbálni, további különböző hálóbeállításokat és hiperparamétereket.