A mesterséges intelligencia és az új kódolási eljárások szerepe a videófeldolgozás fejlődésében

NELSON FRANCISCO, BORDÁS CSABA
MediaKind
Southampton, Egyesült Királyság | Budapest, Magyarország
{nelson.francisco; csaba.bordas}@mediakind.com

Kulcsszavak: videókódolás, mesterséges intelligencia, gépi tanulás, mozgásbecslés, MI-alapú tömörítési technológia

A videótömörítő rendszerek tervezésekor az a fő cél, hogy maximalizáljuk a videó minőségét egy adott bitráta esetében, vagy hogy a lehető legalacsonyabb bitráta mellett érjük el a célzott videóminőséget, mindezt jól meghatározott feldolgozási erőforrások felhasználása mellett. Mivel a gazdasági és környezetvédelmi szempontok gyakran szigorúan korlátozzák az erőforrásokat, a kódolótervezés hagyományosan kodekszakértőkre támaszkodott a heurisztikák és algoritmusok kifejlesztésében, hogy kiválasszák az egyes alkalmazások kódolási eszközeit az előre meghatározott hatékonysági vagy számítási paraméterek szerint. Ezek a heurisztikus módszerek a viszonylag alacsony tömörítési hatékonyságú kódolási eszközök egyszerű letiltásától kezdve a kódolási módok, hivatkozások, mozgásbecslés (Motion Estimation, ME), keresési tartományok vagy blokkméretek közvetlen korlátozásán át terjedhetnek, akár globálisan, akár helyi szinten, a forrás általános jellemzői alapján.
Bár ez a tradicionális megközelítés kiszámítható és következetes kódolási hatékonysági növekedést nyújt meghatározott tartalomtípus esetén, a méretezést megnehezíti annak megértése, hogy az egyes eszközök hogyan befolyásolják az egyes tartalomtípusok tömörítési hatékonyságát, és hogyan lépnek kölcsönhatásba egymással, különösen azért, mert a kódoló összetettsége és a tömörítési hatékonyság közötti kapcsolat nem lineáris.
A cikkbeli elemzés elvégzéséhez a MediaKind kihasználta a gépi tanulás (Machine Learning) lehetőségeit, olyan valós idejű MI-vezérelt kódolási döntéseket hozva ezáltal, amelyek sokkal hatékonyabbnak bizonyultak, mint bármely ember által meghatározott heurisztika vagy algoritmus.

1. Bevezetés

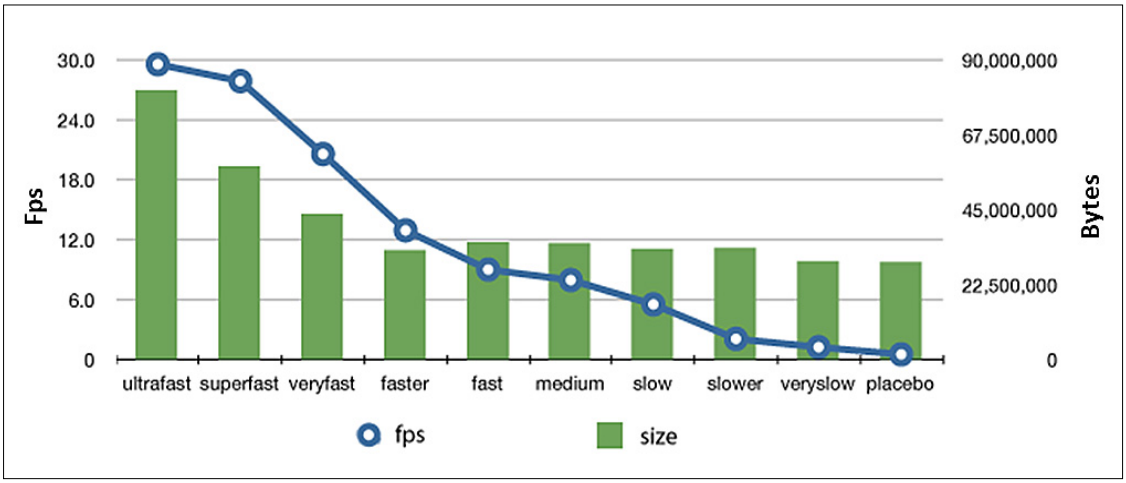
Minden évtizedben a videokodekek új generációja jelenik meg azzal a céllal, hogy elődjéhez képest megduplázza a tömörítési hatékonyságot. Az aktuális fejlesztéseket azonban főként az egyre számosabb és kifinomultabb kódolási eszközök kihasználásával hajtották végre, nem pedig a tömörítési paradigma teljes megváltoztatásával.

Az MPEG-2 [1] például nem használt intra-előrejelzést, amit a H.264 [2] esetében legfeljebb 9 különböző üzemmóddal [3] vezettek be. A HEVC [4] ezen módok számát 35-re emelte [5], a VVC [6] pedig tovább bővíti az intra-előrejelzés fogalmát azáltal, hogy az intra-előrejelzési módok maximális számát 87-re emeli [7]. Hasonlóképpen, az MPEG-2 és a H.264 is 16x16 pixeles makroblokkokat (MBs) [3] határozott meg, amelyeket azóta már felváltott a kódolási faegységek (Coding Tree Units, CTU) általánosabb koncepciója [5]. Ezek HEVC-ben akár 64x64 pixelt, VVC-ben pedig 128x128 képpontot is tartalmazhatnak [6]. Hasonló tendencia figyelhető meg a videokodek szinte minden aspektusában, a mozgásbecslés (Motion Estimation, ME) megnövekedett pontosságánál, a képcsoportstruktúrák (Group of Pictures, GOP) nagyobb rugalmasságánál, vagy akár a rendelkezésre álló hurokban lévő szűrők számában és összetettségében is.

Bár a feldolgozási teljesítményre vonatkozó követelmények növekedését részben teljesítették az állandó hardverfejlesztések, a felbontás és a képkockasebesség egyidejű növekedése azt hozza magával, hogy egyes alkalmazásokhoz nem elegendő kizárólag a hardverfejlesztésekre támaszkodni. A környezeti és gazdaságossági korlátok gyakran közvetlenül korlátozzák a rendelkezésre álló erőforrásokat és a működési költségeket, ami a kodek számítási összetettségének csökkentését igényli a videószállítási folyamat gazdaságilag életképessé tétele és szénlábnyomának minimalizálása érdekében.

Mindezek eredményeként a kódolófejlesztők gyakran szembesülnek azzal, hogy korlátozni kell a kódolási eszközöket annak érdekében, hogy megfeleljenek a célalkalmazás számítási erőforrás-költségvetésének [8,9]. Ez gyakorlatilag azt jelenti, hogy korlátozzuk az előrejelzési módok, referenciaképek vagy mozgáskompenzációs keresési tartományok számát, vagy egyszerűen letiltjuk a specifikáció által meghatározott kódolási eszközök némelyikét.

Fejlesztéseinkben ezért kihasználjuk a gépi tanulás (Machine Learning, ML) és az MI lehetőségeit, hogy optimálisan illesszük a használt kódolási eszközöket a rendelkezésre álló erőforrásokhoz, garantálva, hogy a feldolgozás az egyes alkalmazásokhoz mindig a leghatékonyabb legyen.



1. ábra Profilok összehasonlítása egy 60 mp-es HD-tartalomhoz, különböző x264-es készletekkel kódolva.

2. Tradicionális CODEC-konfigurálás

A videótömörítési mérnökök hagyományosan a tömörítési hatékonyság és a számítási összetettség hányadosának optimalizálása érdekében előre meghatározott paraméterkészletekre támaszkodtak [8,9]. A készletek olyan konfigurációs paramétereket határoznak meg, amelyek korlátozzák a rendelkezésre álló kódolási eszközöket, és amelyek mindegyike eltérő kompromisszumot kínál a számítási követelmény és a bitráta hatékonysága között. Ezt a megközelítést használja például az x264 [8] és az x265 [9], amelyek a H.264 [2] és a HEVC [4] videokodekek nyílt forráskódú implementációi. A beállításkészlet-opciók a placebótól (a legjobb minőség, nagyobb számítási összetettséggel) az ultragyors (alacsony komplexitás, korlátozott tömörítési teljesítmény) konfigurációig terjednek, lehetővé téve a felhasználó számára, hogy a kódoló hatékonyságát a rendelkezésre álló erőforrásoknak megfelelően állítsa be. Az, hogy melyik előre beállított beállítást használja, az alkalmazástól függ, mivel a nagyobb hatékonyság értékét gyakran kereskedelmi tényezők határozzák meg.

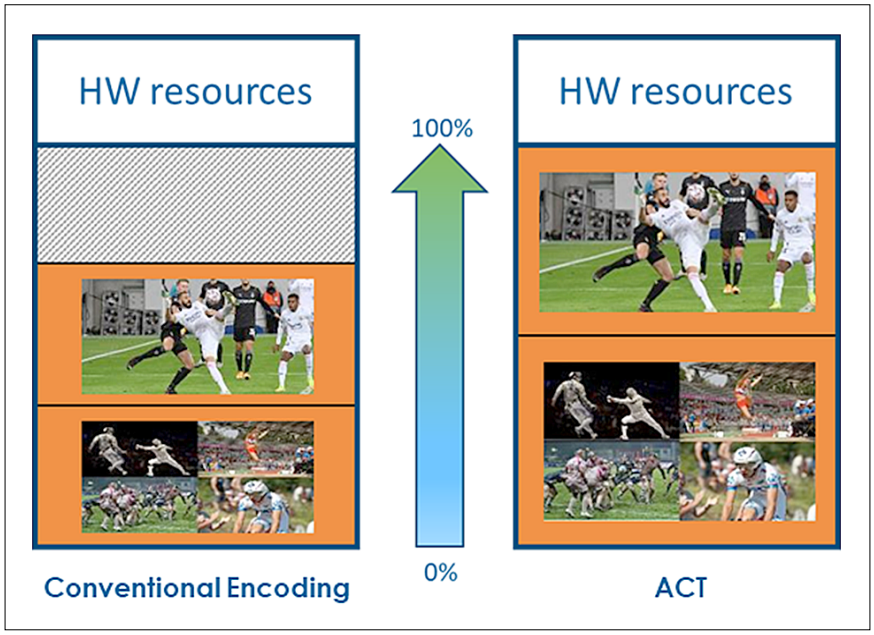
Az *1. ábra* az fps (frame per sec – képkocka/másodperc) számát és az ebből eredő tömörített fájlméretet mutatja egy 60 másodperc hosszú HD-tartalomhoz, amelyet az egyes rendelkezésre álló készletek segítségével kódoltak. Egyértelmű tendencia figyelhető meg mind a tömörítési hatékonyság, mind az erőforrás-követelmények tekintetében: az alacsonyabb profilok nagyobb tömörített fájlt eredményeznek (az alacsonyabb tömörítési hatékonyság eredményeként), de nagyobb számú képkockát tudnak feldolgozni másodpercenként ugyanabban az erőforrásban.

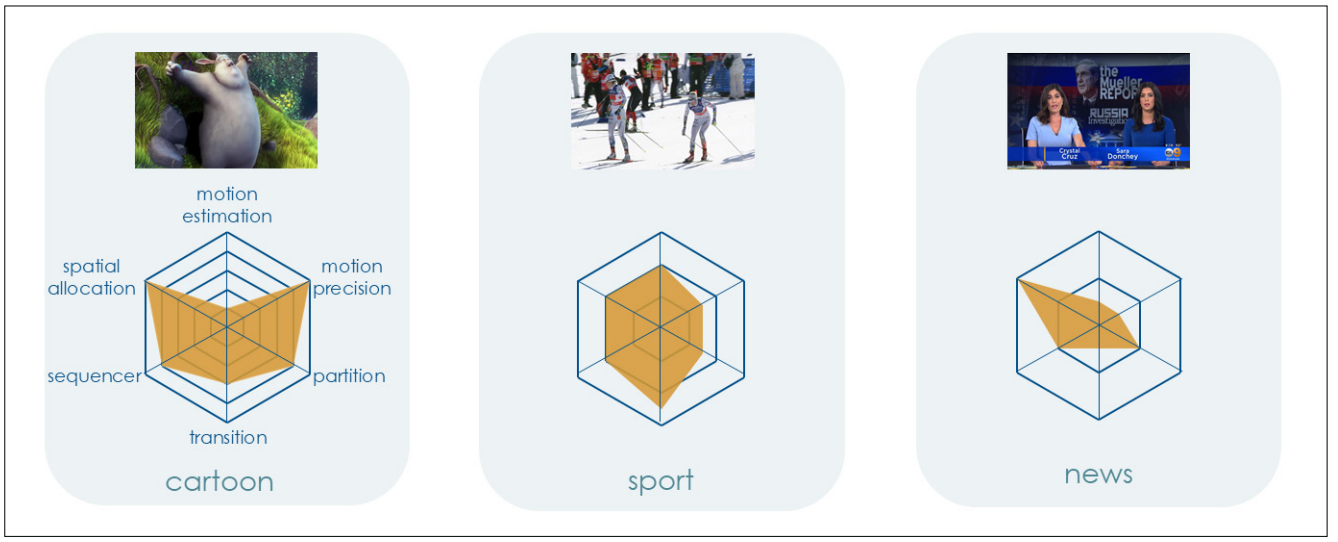
Ebben a példában azonban a „gyorsabb” („faster”) profil jobb tömörítési hatékonyságot ért el, mint néhány más készlet, amelyek több feldolgozást igényelnek. Ez rávilágít az előre beállított megközelítés egyik fő hiányosságára: az összetettebb készletek által hozzáadott extra eszközök és funkciók általában javítják a tartalom tömörítési hatékonyságát, de valójában káros hatással lehetnek bizonyos tartalmakra, a bemeneti forrás jellemzőitől függően.

3. ACT-MI-alapú tömörítési technológia (AI-based Compression Technology)

A konfigurációs, előre beállított megközelítés egyik problémája a kódoló számítási igényének beállításához nyújtott korlátozott lehetőségek. A legösszetettebb készlet, amely még mindig illeszkedik a rendelkezésre álló erőforrásokhoz, elméletileg a legjobb tömörítési hatékonyságot biztosítja, de előfordulhat, hogy az erőforrások jelentős részét nem használja, csak azért, mert a következő, jobb képminőséget nyújtó beállításkészlet csak kismértékben bár, de meghaladja a teljes feldolgozási költségvetést.

2. ábra Az ACT előnye több csatorna estén ugyanabban a rendszerben.





3. ábra Példa a különböző kódolási funkciók hatékonyságára a különböző tartalmakhoz.

Ez a probléma még nyilvánvalóbbá válik, ha több kódolót futtatunk ugyanazon a platformon, és nem működőképes különböző beállításkészletek üzemeltetésére különböző csatornákon. Az ACT-re vonatkozóan megállapított első követelmény tehát az volt, hogy a beállításokat granulárisabbá kell tenni, hogy maximalizáljuk az erőforrás-felhasználást és a tömörítési hatékonyságot *(2. ábra)*.

Az előző szakaszban említett előre beállított megközelítés másik hátránya, hogy az előre beállított készleteket átlagos, általános videóbemenetekre határozzák meg, és mind a komplexitás, mind a tömörítés hatékonysága jelentősen eltérhet a tartalomtípusok között. A különböző videóbemeneti jellemzők közvetlenül befolyásolják az egyes készletek által meghatározott kódolási eszközök hatékonysági előnyeit.

Amint azt a *3. ábra* mutatja, a rajzfilmek általában részletes textúrát és éles éleket mutatnak be, ami azt jelenti, hogy a pontos térbeli elosztás elengedhetetlen ahhoz, hogy pontosan ábrázolja a térbeli komplexitás variációit a képen. Továbbá, míg a mozgás általában könnyen nyomon követhető és megjósolható, tehát nem igényel túl sokat a mozgásbecsléstől, nagy mozgási pontosságra és nagyon pontos helymeghatározásra van szükség ahhoz, hogy elkerüljük az észrevehető kódolási hibákat.

Egy másik alkalmazást nézve, a térbeli kiosztás nem olyan fontos a sporttartalmakban, ahol a gyors kameramozgás, a gyors zoom és a térbeli és időbeli maszkolási hatások okozta elmosódás kevésbé észrevehető a nézők számára. Maga a mozgás azonban összetett és nehezen nyomon követhető az elzáródások, vágások és a világításváltozások miatt. Az olyan eszközök, mint a súlyozott előrejelzés, az extra erőforrások kiosztása a szekvenáló processzorba a leghatékonyabb GOP-struktúra meghatározásához, vagy több hivatkozás felvétele segíthet a mozgásbecslésben, és nagyon pozitív hatással lesz az általános tömörítési hatékonyságra.

A hír- és a stúdiótartalmaknál fontos a térbeli kiosztás, különösen átfedő grafikák esetén, de a mozgás könnyen nyomon követhető, tekintettel a korlátozott kameramozgásra és a jól meghatározott átmenetekre.

Annak érdekében, hogy jobban megértsük, hogyan befolyásolják a különböző tartalomtípusok az erőforrás-kiosztást, és határozzák meg a különböző típusú kódolási eszközök hatását a tömörítési hatékonyságban, az eszközöket kategóriákba csoportosították a közvetlenül érintett kodekterület szerint *(1. táblázat)*.

Az adaptív kvantálás és a sebességszabályozás például a bitráta-kiosztási stratégia részét képezi, míg a jelenetvágás észlelése és a GOP-tervező a szekvenálási kategóriába tartozik. Hasonlóképpen, míg a mozgásvektor finomítása a mozgáskompenzációs pontossági kategóriába tartozik, maga a mozgásbecslés, a referenciakezelés és a súlyozott előrejelzés a mozgásbecslési stratégia kategóriájába tartozik. Figyelembe kell vegyük, hogy függőség lehet a különböző kategóriákból származó eszközök között, mivel például a HEVC CU (Cod-

1.táblázat Kódolási eszközkategóriák.

Címke	A funkció leírása
ME	Mozgásbecslési stratégia
MCP	Mozgáskompenzálás pontossága
IPS	Predikción belüli keresés
PIPA	Kép particionálása
MD	Mód döntési stratégia
TQR	Átalakítás és kvantálás finomítása
BA	Bitráta-kiosztási stratégia
SEQ	Szekventálás
TM	Átmenet kezelése
PPE	Előfeldolgozás és becslés

4. ábra A vizsgált tartalmak.

ing Units) méreteinek korlátozása nemcsak a kép particionálási kategóriáját befolyásolja közvetlenül, hanem az átalakítás és a kvantálás finomítását is, mivel egyes átalakításméretek elérhetetlenné válnak. Az egyszerűség érdekében azonban elvonatkoztathatunk ezektől a függőségektől, anélkül, hogy ez veszélyeztetné az eredményeinket.

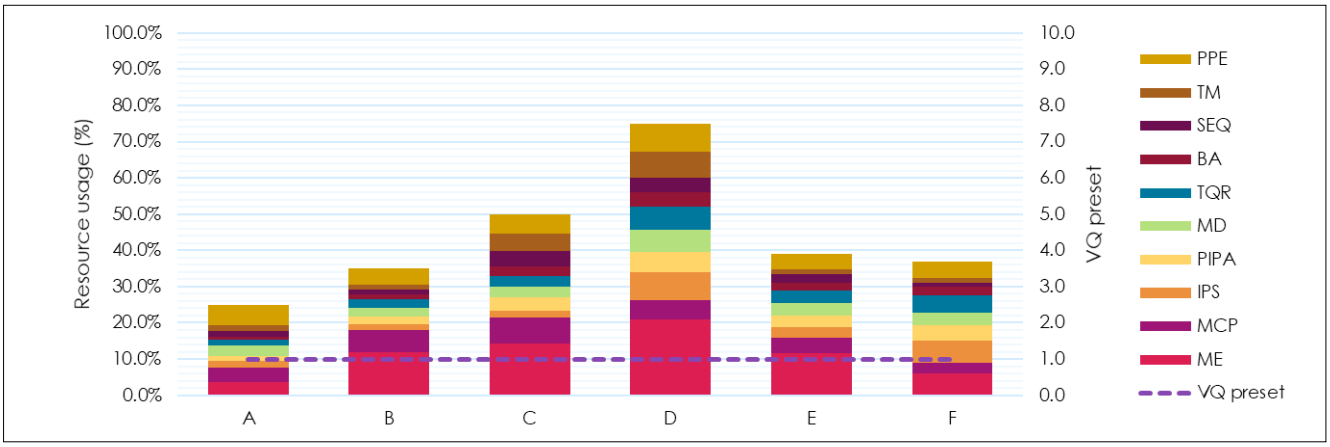
A *4. ábrán* összefoglalt, különböző térbeli és időbeli komplexitású tartalmak egy adott előre beállított videóminőségi értékre kódolásával az *5. ábra* egyértelműen mutatja az erőforrás-felhasználás és a bemeneti videó jellemzői közötti függőséget. Ebben a példában a *D* tartalomhoz közel háromszor több feldolgozásra volt szükség, mint az *A* tartalomhoz, amikor ugyanabban az előre beállított készletben kódolták őket. Emellett az egyes kódolási eszközkategóriák relatív erőforrás-felhasználása jelentősen változik a bemeneti videó jellemzőitől függően.

Az előre beállított megközelítés hatékonysági korlátai egyértelműen láthatóak az ábrán, mivel adott erőforrás-költségvetés mellett itt a *D* tartalom határozza meg a maximálisan használható beállításokat, ami elpazarolt erőforrást jelent bármely más tartalom esetén. Felmerül egy másik kérdés is: valóban a *D* tartalom a legrosszabb forgatókönyv? Ha a kódoló bemenetére a *D* tartalomnál is összetettebb videó kerül, átlépjük az erőforrás-költségvetést, ami katasztrofális következményekkel járhat, például át kell ugrani egy keretet, vagy összeomlik a kódoló. Ez gyakran azt jelenti, hogy az előre meghatározott beállítást még konzervatívabb módon kell kiválasztani, bizonyos biztonsági tartalék méretezésével, tovább növelve az erőforrás-pazarlást.

Az erőforrás-felhasználás figyelésével és a kódoló konfigurációjának dinamikus és valós idejű beállításával több eszköz engedélyezhető, amikor több erőforrás áll rendelkezésre. A gyakorlatban ez azt jelenti, hogy jobb videóminőségű előre beállított készletek használhatóak, ha a tartalom kevesebb erőforrást igényel. Ez a rendelkezésre álló feldolgozási erőforrások hatékonyabb felhasználását eredményezi, amint azt a *6. ábra* mutatja.

A korábbi megfigyelések alapján tehát meghatározzuk az optimalizálási problémát: azon kódolási eszközök és beállítások készletének megtalálása, amelyek maximalizálják a VQ-t (videóminőséget) egy adott tartalomhoz, egy adott t vagy $t+\Delta t$ pillanatban, miközben megfelelünk a valós idejű kódolás korlátozásának.

5. ábra Erőforrás-felhasználás eszközkategória szerint a különböző tartalomosztályokhoz.



Meghatározások:

• C a tartalomjellemzők vektora (tömörítési formátum, térbeli- és mozgáskomplexitások, gradienselemzés és időbeli variációs metrikák, online becsléssel a bemeneti forrás előzetes elemzése alapján).

• $p(C) \in \{p_1(C), .. , p_M(C)\}$ a kódolási előre beállított vagy kódolási paraméter-kombinációk listája a tartalomjellemzők függvényében, úgy, hogy:

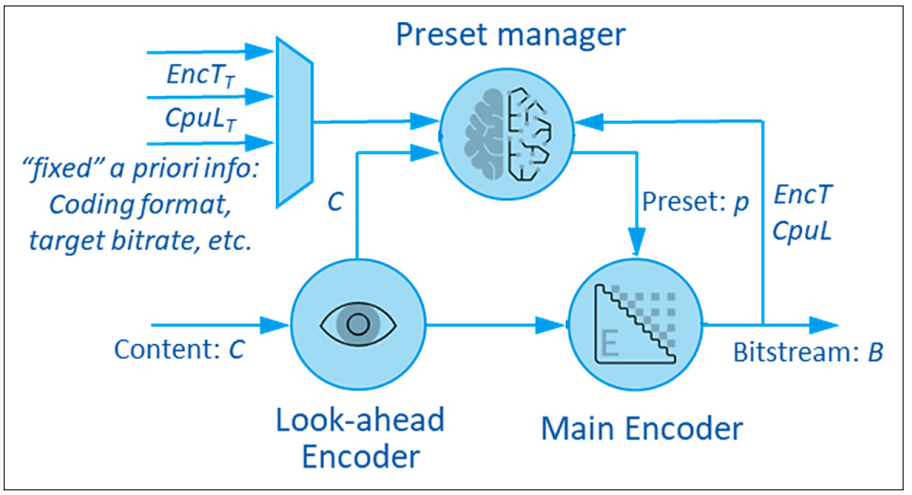
– A kódolási készlet a kódolási eszközszintek készlete/vektora; minden olyan kódolási eszközzel, amelyről feltételezzük, hogy legfeljebb *N*-szintű kódolási hatékonysággal/összetettségi kompromisszumokkal rendelkezik;

– A készletek listája a leggyorsabb/legalacsonyabb VQ-tól (1. készlet) a leglassabb/legjobb VQ-ig (előre beállított M) van meghatározva.

• *EncT* kódolási idő (ms/keret).

• *X*: kódolási futási célidő a kért kimeneti keret kódolási valós idejű küszöbértékének %-ában, $EncT_T = X * EncT_{realtime}$, $ahol\ 0 < X \leq 1$.

• *CpuL*: CPU/rendszer terhelésmérés.



7. ábra Az ACT-vezérlő blokkdiagramja.

Bármilyen *t* időpillanatban a C vektorjellemzőkkel rendelkező tartalom és egyéb a priori ismeretek, mint például a cél-bitráta, célhardver, R_T, stb. mellett meg akarjuk találni az optimális $p^*(C) \in \{p_1(C), .. , p_M(C)\}$ értékkészletet, amely maximalizálja a VQ-t az adott korlátok között: $EncT_T$ kódolási idő és $CpuL_T$ CPU terhelési cél:

$$\forall t, \begin{cases} p^*(C, \{HW, R_T, ..\}) = \max\limits_{p \in \{p_1 .. p_M\}} VQ\big(p(C, \{HW,\ R_T, ..\})\big) \\ EncT\big(p(C, \{HW, R_T, ..\})\big) = EncT_T \\ CpuL\big(p(C, \{HW, R_T, ..\})\big) \leq CpuL_T \end{cases}$$

Mivel a fő cél egyszerű módszer valós idejű futtatása volt, anélkül, hogy túl sok olyan erőforrást fogyasztanánk, amelyeket egyébként maga a kódoló használhatna, a C meghatározásánál a „lookahead”-kódolóban (a hagyományos kódoló prediktív része tipikusan 10-20 keret pufferelésével operál) a már kiszámított metrikákat használjuk a következőképpen:

• Intra-komplexitás vagy térbeli aktivitás:
$I = \sum_i^N \sum_j^M |x_{ij} - \mu|$, és $\mu = \frac{1}{NM} \sum_i^N \sum_j^M x_{ij}$.

• Inter-komplexitás:
$E = \sum_i^N \sum_j^M |x_{ij} - \hat{x}_{ij}|$,
ahol $\hat{x}_{ij}$ az x_{ij} mozgásbecsült blokk.

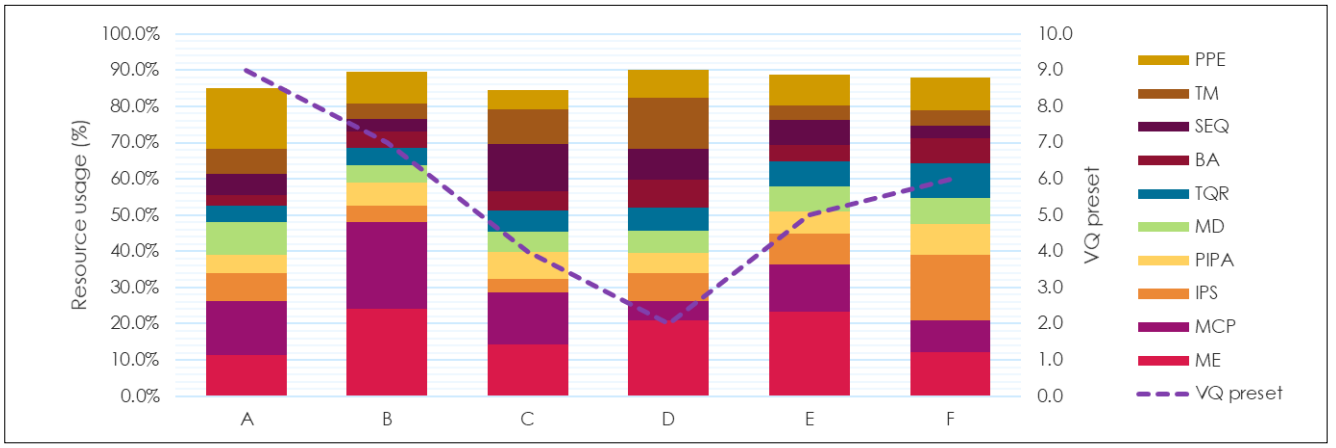
• Színátmenetek blokkolása:
$R = \sum_i^{N-1} \sum_j^{M-1} |x_{i+1,j+1} - x_{i,j}|$.

• Mozgásvektor költsége és entrópia.

Ezeknek a metrikáknak az a priori információkkal való kombinációja, például a kódolási formátum és a cél-bitráta lehetővé teszi a használandó konfigurációs paraméterek első becslését, a visszacsatolási hurok pedig korrekciós információkat nyújt arról, hogy a kódoló futásideje és a rendszer terhelése hogyan változik az idő múlásával, a további képkockánkénti korrekció végrehajtása érdekében, ahogyan azt a *7. ábra* mutatja.

Több kódolás futtatásával, a különböző kódolási eszközök konfigurációs paramétereinek a sokféle bemeneti tartalomhoz igazodó beállításával ezután neurális hálózat (Neural Network, NN) segítségével kinyerhető és meghatározható a „lookahead”-metrikák és az optimális

6. ábra Előre beállított készletek adaptációja tartalom szerint.



konfigurációk közötti korreláció. Az NN ezután felhasználható a bemeneti kép jellemzőinek megfelelő optimális konfiguráció valós idejű beállításainak végrehajtására is.

Az algoritmust mind a H.264, mind a HEVC esetében megvalósították és tesztelték. Ugyanazt a C tartalomjellemző-vektort használták, de a $p(C)$ konfigurációs paraméterek természetesen különböznek, mivel az egyes kodekeken elérhető eszközök nem azonosak.

A számítási erőforrások rögzítése mellett a javasolt módszer 18%, illetve 19%-os BD-SSIM (Bjøntegaard Delta Structural Similarity) nyereséget ért el a H.264 és a HEVC esetében, összehasonlítva egy rögzített előre beállított konfigurációval. A referenciaérték meghatározásakor a leghatékonyabb előre meghatározott beállítást használták, amely valós idejű kódolást garantál a rendelkezésre álló keretek között egy nagy, különböző tartalmakból álló tesztkészletben – míg az ACT minden képkockához beállította a kódolási paramétereket. Az eredményeket egyetlen csatornára számították ki, de amint azt korábban kifejtettük, a nyereségek még jelentősebbek lehetnek több csatornát futtató rendszerek esetében.

4. MI vezérelt HEVC CU felosztási döntések

Míg az MPEG-2 és H264 16x16 pixel méretű blokkegységeket fogadott el, amelyeket MBs-nek (Macroblocks) [3] neveznek, a HEVC új, általánosabb blokk-entitást vezetett be, amelyet CTU-nak (Coding Tree Unit, Kódolási Faegység) neveznek [8]. A CTU-k mérete 64x64, 32x32 vagy 16x16 pixel lehet, ami hierarchikusan tovább oszlik CU-kra (Coding Unit, kódolási egységek), az eredeti CTU mérettől 8x8 képpontig. Ez növeli a kódolás hatékonyságát azáltal, hogy lehetővé teszi az előrejelzések és a méretek jobb beállítását a tartalom helyi jellemzőihez, a számítási igények jelentős növekedésének rovására.

A valószínűleg használt blokkméretek pontosabb előrejelzése a kódolás előtt azt jelenti, hogy az összes lehetséges opciónak csak egy részhalmazát kell kiszámítania a fő kódolónak – hatalmas potenciális erőforrás-megtakarítással –, mivel ez a blokkméret-optimalizálási hurok jelentősen hozzájárul a kodek összetettségéhez.

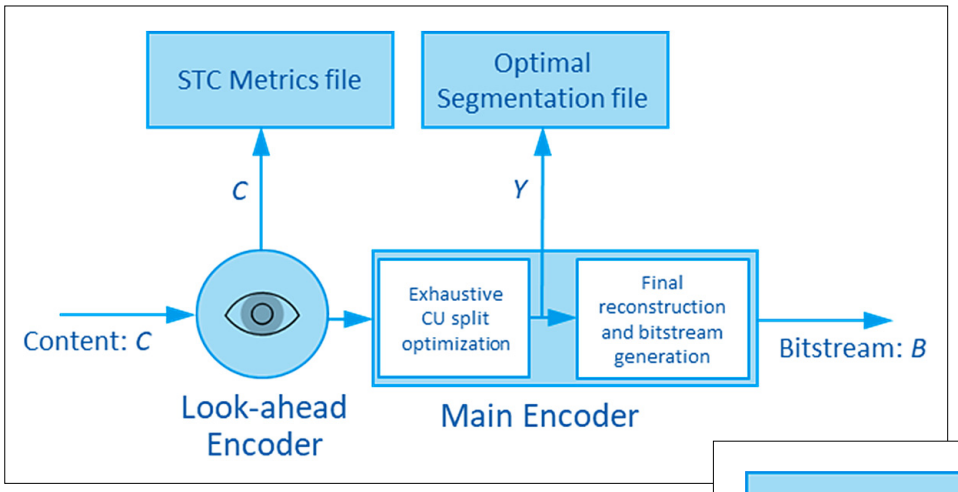
Nem meglepő tehát, hogy számos kutatás foglalkozott ezzel a problémával, és ez az a terület, ahol a gépi tanulás a legígéretesebb eredményeket hozta. Momcilovic és társai [11] olyan módszert javasoltak, amely akár 65%-os kódolási időcsökkenést ért el a kódolási ráta ingadozásának elhanyagolható növekedésével. Nem igényel betanítást, rendkívül adaptív algoritmust eredményez, amely dinamikusan reagál a videóbemenet változásaira. Egy másik tanulmány szerzői gépi tanuláson alapuló módszert javasoltak olyan funkciók felhasználásával, amelyek leírják a CU-statisztikákat és a képernyőtartalom al-CU-homogenitását [12]. A szerzők átlagosan 36,8%-os komplexitáscsökkenést értek el, mindössze 3,0%-os bitráta-növekedéssel. Chen és társai [13] egy gyors kódolási egységet (CU) javasoltak a HEVC-n belüli kódoláshoz egy mesterséges neurális hálózat (ANN) és egy támogató vektorgép (SVM) felhasználásával. Módszerükben a gépi tanulás szisztematikus megközelítést biztosított a korai CU-megosztás vagy -megszüntetés gyors algoritmusának kifejlesztéséhez a kódoláson belüli számítási komplexitás csökkentése érdekében. A Convolutional-Neural-Network (CNN) használata talán a legsikeresebb megközelítés volt. A szerzők [14]-ben egy kontrasztnyereség-ellenőrzési modellt javasolnak, amely megpróbálja megragadni a felismerhető struktúrák hatását a torzítás láthatóságára. Liu és társai [15,16] egy bonyolult neurális hálózaton alapuló gyors algoritmust javasoltak, hogy minden CTU-ban legalább kettővel csökkentsük a CU-partíciósmódok számát. Ez a módszer teljes sebességtorzulás-optimalizálást (RDO) használ. Az algoritmus az intra-kódolási idő 63%-át takarította meg átlagosan 2,66%-os BD-BR (Bjøntegaard Delta Bit-Rate) növekedési költségen. Javasolták a CNN-ek használatát [17] is, hogy megjósolják a 32x32 pixeles CTA-k 90%-os pontossággal történő felosztását.

A CNN-ek előnye, hogy lehetővé teszik a modell számára, hogy hatékonyan kinyerje a leghasznosabb funkciókat a bemeneti adatokból, ami akkor fontos, ha a hangsúly a legjobb pontosság elérésén van. A CNN-ek azonban számítási szempontból költségesek, és a GPU-k használatát igénylik a valós idejű kódoláshoz alkalmas átviteli sebesség eléréséhez. Ez azt jelenti, hogy nem biztosítanak mérhető teljesítménynövekedést a valós idejű kódoláskor, mivel a kódoló összetettsége már viszonylag alacsony [17].

A javasolt megközelítés középpontjában a modell pontossága és összetettsége közötti egyensúly megteremtése állt, tekintettel a valós idejű alkalmazásokban kitűzött célra. Így ahelyett, hogy CNN-eket használnánk a CU felosztásának megállapításához használt videójellemzők kinyerésére, újra felhasználtuk a „lookahead”-kódolóban már elérhető metrikákat. Ez a megközelítés korlátozhatja az előrejelzések pontosságát, de nem igényli a GPU használatát, így a módszer valóban általános és platformfüggetlen. Ugyanazokat a „lookahead”-metrikákat használjuk, mint az ACT-ben, nevezetesen a térbeli aktivitást, az inter-komplexitást, és az egyes 16x16 blokkokhoz kiszámított gradienseket, egy alul-mintavételezett képen. A CU QP (Compute Unit Quantization Parameter) és a kerettípus is szerepel a következtetéshez használt adatokban.

Különböző hálózati architektúrákat vizsgáltunk, különböző számú és kiterjedésű rétegekkel. A következtetés pontossága és a költségek közötti kompromisszumként egy 4 teljesen összekapcsolt réteggel rendelkező NN-t használtunk. Az első 3 rétegnek 50 neuronja van, amelyek ReLU (Rectified Linear Unit) aktiválási funkciót használnak, míg az utolsó rétegnek 21 neuronja van, és ez sigmoid aktiválási funkciót használ. Ez a hálózat átlagosan több mint 92%-os pontosságot ért el a CU-k 64x64 és 16x16 közötti felosztása esetén (a 8x8 nem osztható tovább).

Az NN-betanítást offline, GPU-val végezték egy több mint kétezer videósorozatból álló reprezentatív képzési



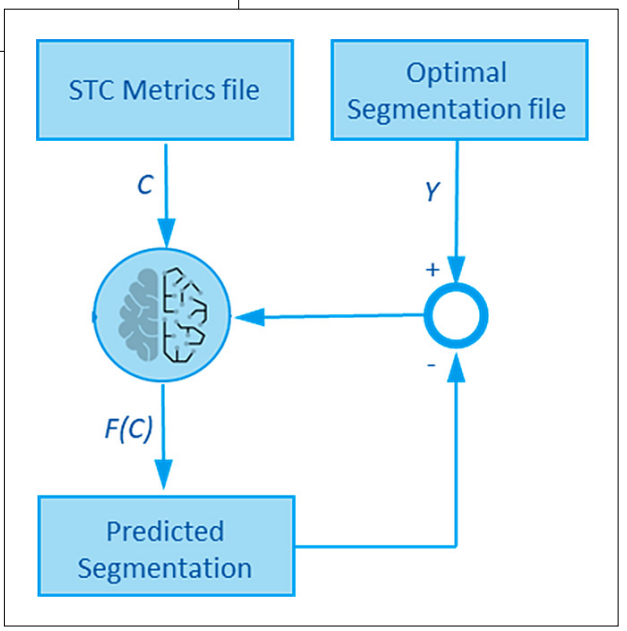
8. ábra
NN adatkészlet-generálás.

készleten. A bemeneti videót több QP-ponton kódolták, amelyek a teljes QP-tartományt lefedik és teljes RDO-keresést használnak, ahol az összes CU-méret engedélyezve van. Mind a lookahead-metrikák, mind az egyes CTU-khoz származó optimális szegmentációs jelzők naplózva voltak fájlokba mentve, amint azt a *8. ábra* mutatja.

Felügyelt betanítási megközelítést alkalmaztak, a *C* mérőszámokat az NN-be táplálták, hogy *F(C)* következtetést hajtsanak végre a CTU felosztásához. A kódolóból a teljes RDO kereséssel nyert optimális szegmentációt alapvetésként használják, amely lehetővé teszi a következtetett és az alapfelvetés közötti bináris keresztentrópia-vesztés meghatározását:

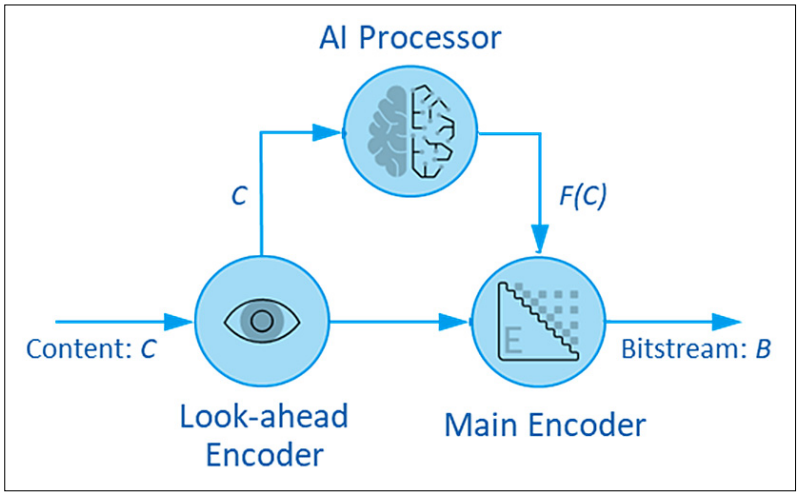
$$L = -\sum_{n} w_n \left[y_n \log \sigma(F(c_n)) + (1 - y_n)\left(1 - \log \sigma\left(F(c_n)\right)\right)\right]$$

Ez a megközelítés feltételezi, hogy minél közelebb van az NN döntése a teljes optimalizálási eredményekhez, annál optimálisabb lesz a megoldás. A súlygradiensek visszaemelése lehetővé teszi a sztochasztikus gradiens süllyedési optimalizálását pillanattal [18] az NN-súlyok gradiensei felett, interaktív módon csökkentve a teljes veszteséget. A *9. ábrán* felvázoltuk a betanítási folyamatot. A betanítási adatokat képzési és érvényesítési készletekre osztották, az adatpontok 80%-ával, illetve 20%-ával.



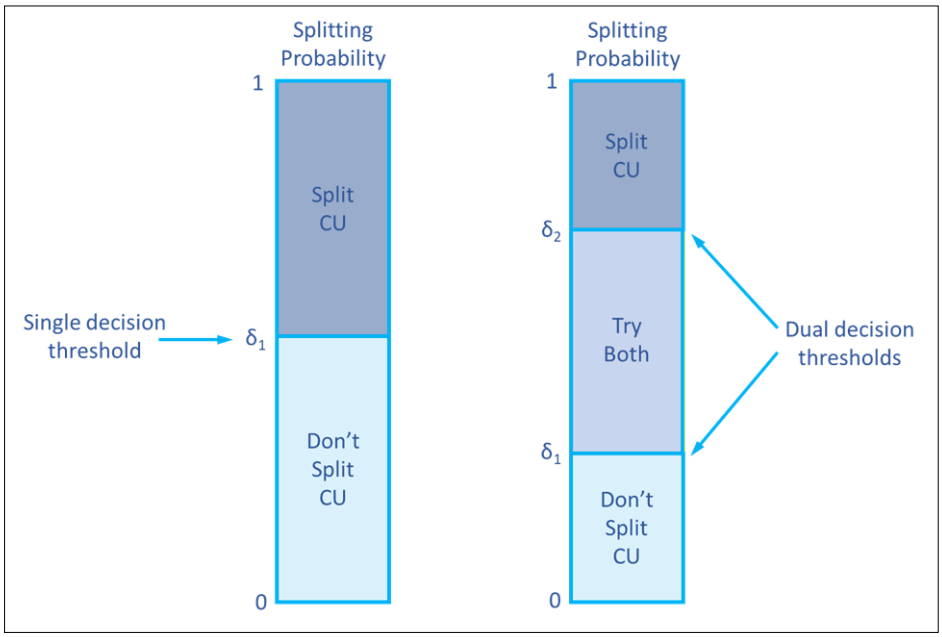
9. ábra
NN-betanítási folyamat.

10. ábra NN-következtetés a valós idejű kódolóban.



Miután az offline betanítás befejeződött, az NN-súlyokat a kódolóba táplálták, hogy a döntés valós időben elvégezhető legyen, egyszerűen a „lookahead”-kódoló adatainak felhasználásával, lásd a *10. ábrát*. Mivel NN viszonylag kicsi, többnyire lineáris rétegekre támaszkodik, amelyek hatékonyan megvalósíthatók SIMD-sel, azt jelenti, hogy a döntés nagyon alacsony költséggel elvégezhető a CPU-ban. Fontos, hogy ezzel a megközelítéssel a modell könnyen frissíthető minden alkalommal, amikor az offline betanítás olyan súlyokat eredményez, amelyek jobb pontosságot biztosíthatnak.

A módszerben alkalmazott valószínűségi megközelítésnek az az előnye, hogy skálázhatóvá teszi azt. Tulajdonképpen minden CTU esetében a következtetés határozza meg annak valószínűségét, hogy az egyes CU-kat szegmentálják-e vagy sem. A nullához közeli *F(C)* valószínűségek azt jelentik, hogy a CU szegmentálásának K+F költsége nem valószínű, hogy alacsonyabb lesz, mint a nem szegmentálás költsége, míg az 1-hez közeli valószínűségek azt jelzik, hogy a CU szegmentálása valószínűleg alacsonyabb RD-költséget eredményez.



11. ábra Egyedüli és kettős döntési küszöbértékek

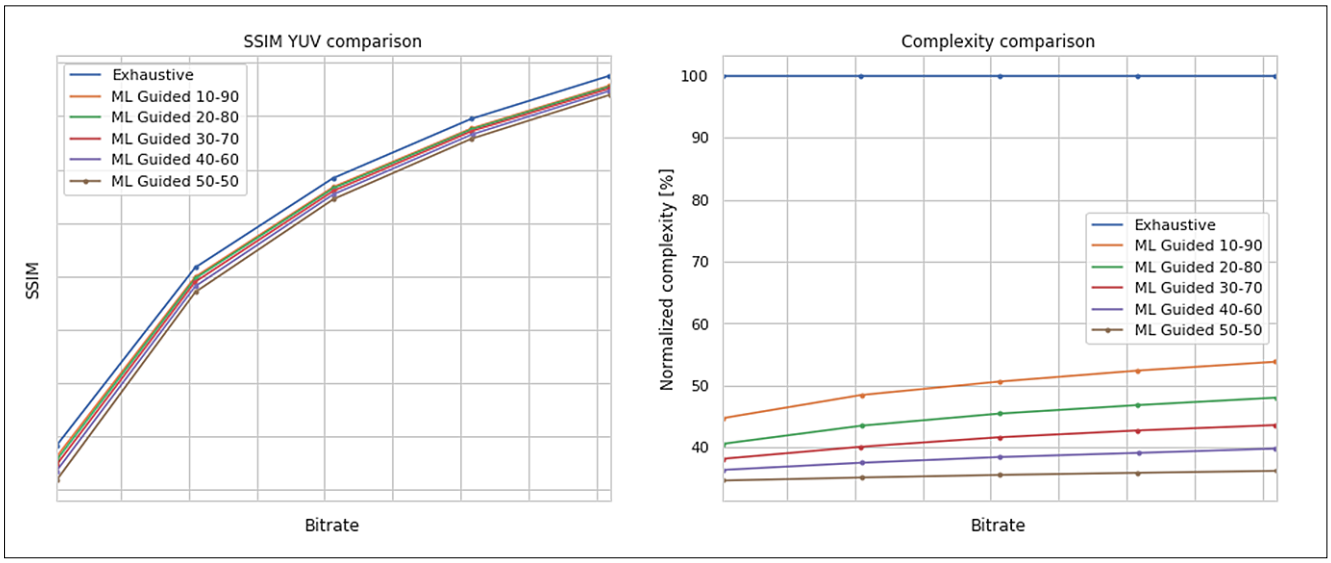
A legnagyobb számítási megtakarítást egyetlen δ_1 küszöbérték meghatározásakor érjük el. A CU-t akkor osztjuk fel, ha az $F(C) > \delta_1$, és nem osztjuk meg, ha az $F(C) < \delta_1$, tehát ebben az esetben minden CU-ra csak az egyik felosztási lehetőséget kell kiszámítani. A 0,5-höz közeli valószínűségek azonban nagy bizonytalanságot jelentenek, amely könnyen egy nem optimális döntéshez vezethet, ami ezek után befolyásolhatja a kódoló tömörítési hatékonyságát. Ez a probléma megkerülhető két különböző δ_1 és δ_2 küszöbérték meghatározásával, ahol $\delta_1 < F(C) < \delta_2$ esetén mind a felosztási, mind a nem felosztási lehetőségeket tesztelve viszonylag alacsony számítási költséggel csökkenthető az optimálistól eltérő döntés kényszerítésének valószínűsége *(11. ábra)*.

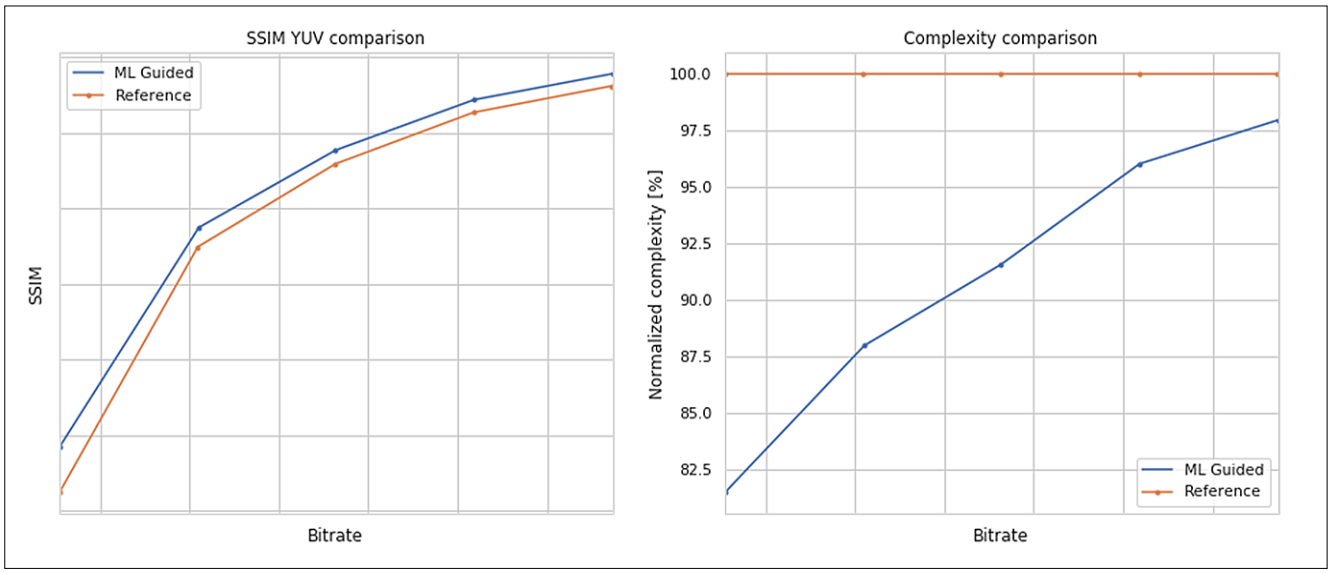
Az intervallum méretének növelése tömörítési hatékonyságot növelő döntésekhez vezet (kevesebb kockázatot vállalva több mód tesztelésével), míg az intervallum szűkítésével a döntéseket a számítási megtakarítások felé toljuk (lehetővé téve a kódoló számára, hogy több kockázatot vállaljon és több módot dobjon el). A rendelkezésre álló erőforrások visszacsatolási hurkának megtartásával ez az intervallum valós időben, az ACT-vel párhuzamosan módosítható.

Az alsó,*12. ábrán* összehasonlítjuk a képminőséget és a kódoló összetettségét, különböző valószínűségi küszöbértékek használata esetén. A grafikonokon látható eredmények több 3840x2160@60fps szekvenciával rendelkező tesztkészletek átlagát mutatják, különböző típusú tartalmak esetén, a sporttól a filmekig.

A képminőséget az SSIM-mel (Structural Similarity Index Measure) mérik. Hogy elvonatkoztassunk a használt kódolási platformtól, a kódolás összetettségét a kimerítő keresés (Exhaustive Search) összetettsége alapján normalizáltuk. Ez azt jelenti, hogy a 100% ugyanolyan összetettségnek felel meg, mint a teljes keresés, míg az 50% azt jelenti, hogy az új kódoló a teljes kereséshez szükséges számítás felét igényli (a feldolgozási erőforrások feleakkorák ugyanazon keretsebesség eléréséhez, vagy ugyanazon a hardverplatformon feleződik a képkockaidő). Az ML-vezérelt 50-50 görbe esetén $\delta_1=\delta_2=0,5$, míg az ML-vezérelt 10-90 esetén $\delta_1=0,1$ és $\delta_2=0,9$ értékeknek felelnek meg. Az összes konfigurációs paraméter változatlan marad, és az ML vezérelt konfigurációk összetettsége magában foglalja az NN következtetését és a kódolást, a referenciaértékkel való közvetlen és korrekt összehasonlításhoz.

12. ábra Tömörítési teljesítmény és kódoló komplexitás összehasonlítása különböző valószínűségi küszöbértékekkel.





13. ábra Videominőségi és számítási összetettségi eredmények egy UHD 60 fps tesztkészlethez.

Ahogy az várható volt, a legjobb videóminőséget a teljes kereséssel érik el. Agresszívebb ML-vezérelt 50-50 esetén akár 3 csatornát is kódolhatunk ugyanazzal a számítási kapacitással, míg a BD-SSIM csökkenés mindössze 5,5% körül van. A bizonytalansági intervallum növelése lehetővé teszi a tömörítési hatékonysági büntetés csökkentését, miközben továbbra is 50% feletti átlagos normalizált összetettségi csökkenést érünk el. Ez azt jelenti, hogy legalább 2 csatorna futtatható ugyanazon a hardverplatformon, és csak szerény 2%-kal csökken a BD-SSIM.

A *13. ábra* egy valós idejű alkalmazás eredményeit mutatja be, ugyanazzal a 3840x2160@60fps tesztkészlettel. A nagy felbontás és keretsebesség komoly hardvererőforrás-igényeket támasztanak valós idejű tömörítés eléréséhez, és még a nagy kapacitású, drága szervereken is csökkenteni kell a kodek eszközkészletét egy tipikus 1080p konfigurációkhoz képest. Az ML-vezérelt CU-felosztás által nyújtott számítási megtakarítások újra felhasználhatók más, korábban letiltott kódolási eszközök engedélyezéséhez, ami több mint 10%-kal javítja a kódoló tömörítési hatékonyságát, miközben a teljes számítási igények alacsonyabb maradnak, mint az eredeti konfigurációban. Vegyük észre, hogy a számítási megtakarítások jelentősebbek alacsonyabb bitráta esetén, mivel az alacsonyabb bitráta általában nagyobb CUs méreteket eredményez, lehetővé téve, hogy ugyanazon bizonytalansági intervallumban több felosztási lehetőséget ugorjunk át.

5. Összefoglalás

Ebben a tanulmányban gépi tanulási ML-technikák használatát javasoltuk a tömörítési hatékonyság és számítási komplexitás hányados javítására a valós idejű alkalmazásokhoz használt hagyományos videokódolókon.

A javasolt algoritmusok lehetővé tették a hatékonyabb feldolgozási erőforrások felhasználását a kódolási lehetőségek dinamikus szűkítésével és a kodek paraméterek módosításával. Egyes esetekben akár 50%-os számítási kapacitás megtakarítást mérhetünk hasonló tömörítési arányoknál, vagy akár 20%-os tömörítési hatékonyságnövekedést egyenértékű számítási követelmények esetében.

A kodek más területei, például az Intra/Inter/Skip döntések is használhatnák egy ilyen megközelítés előnyeit, a jövőben ezeket is meg szeretnénk vizsgálni.

Hivatkozások

[1] H.262: Information technology – Generic coding of moving pictures and associated audio information: Systems. www.itu.int. ITU. 2021-06.
[2] H.264: Advanced video coding for generic audiovisual services. www.itu.int. ITU. 2021-08.
[3] T. Wiegand, G.J. Sullivan, G. Bjontegaard and A. Luthra, „Overview of the H.264/AVC video coding standard," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, no.7, pp.560–576, July 2003, doi: 10.1109/TCSVT.2003.815165.
[4] H.265: High efficiency video coding. www.itu.int. ITU. 2021-08.
[5] G.J. Sullivan, J. Ohm, W. Han and T. Wiegand, „Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, no.12, pp.1649–1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.
[6] H.266: Versatile Video Coding. www.itu.int. ITU. 2020-08.
[7] J. Pfaff et al., „Intra Prediction and Mode Coding in VVC", in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 31, no.10, pp.3834–3847, Oct. 2021, doi: 10.1109/TCSVT.2021.3072430.
[8] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, „Block Partitioning Structure in the HEVC Standard," in IEEE Transactions on Circuits and Systems for Video Technology, pp. 1697–1706, 2012.

[9] Loren Merritt et al,
„X264: A High Performance H.264/AVC Encoder”, 2006.
[10] D. Silveira, M. Porto and S. Bampi,
„Performance and energy consumption analysis of the X265 video encoder,” 25th European Signal Processing Conference (EUSIPCO), 2017, pp.1519–1523,
doi: 10.23919/EUSIPCO.2017.8081463.
[11] S. Momcilovic, N. Roma, L. Sousa, and I. Milentijevic,
„Run-Time Machine Learning for HEVC/H.265 Fast Partitioning Decision,”
in IEEE International Symposium on Multimedia, 2015.
[12] F. Duanmu, Z. Ma, and Y. Wang,
„Fast CU partition decision using machine learning for screen content compression,”
in IEEE International Conference on Image Processing (ICIP 2015), 2015, pp.4972–4976,
doi: 10.1109/ICIP.2015.7351753.
[13] Z.-Y. Chen, J.-T. Fang, Y.-C. Liu, and P.-C. Chang,
„Machine Learning-based Fast Intra Coding Unit Depth Decision for High Efficiency Video Coding,”
Journal of Information Science and Engineering, 2016.
[14] M.M. Alam, T. Nguyen, M. Hagan, and D. Chandler,
„A perceptual quantization strategy for HEVC based on a convolutional neural network trained on natural images”
2015, p.959918,
doi: 10.1117/12.2188913.
[15] Z. Liu, X. Yu, S. Chen, and D. Wang,
„CNN oriented fast HEVC intra CU mode decision,”
in IEEE International Symposium on Circuits and Systems, 2016.
[16] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang,
„CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network,”
in IEEE Transactions on Image Processing, 2016.
[17] J. Wenchan, Y. Ming, X. Ying, and L. Zhigang,
A Machine Learning Method for Optimizing Partition of Prediction Block in Coding Unit in H.265/HEVC, (2020).
doi: 10.4108/eai.27-8-2020.2297985.
[18] Kingma, Diederik and Ba, Jimmy,
Adam: A Method for Stochastic Optimization, (2020).
International Conference on Learning Representations.

A szerzőkről

NELSON FRANCISCO a MediaKind kutató mérnöke 2008-ban szerzett elektronikai mérnöki mesterdiplomát a Tras-os-Montes e Alto Douro Egyetemen (Portugália) és 2012-ben PhD fokozatot jelfeldolgozás témakörben a Rio de Janeiro Szövetségi Egyetemen. 2007–2012 között a Portugál Távközlési Intézettel (Portuguese Telecommunication Institute) működött együtt több, a kép- és videófeldolgozásra összpontosító kormányzati finanszírozású projektben, majd 2013-ban csatlakozott a Poznani Műszaki Egyetem multimédiás távközlési csoportjához, mint látogató posztdoktori kutató. 2010–2012 között a Leiriai Műszaki Egyetem adjunktusa volt Portugáliában. 2013-ban került az Ericssonhoz kutatómérnökként, ahol online és offline hardver és szoftverkódolási eljárások alapalgoritmusainak fejlesztésén dolgozott. 2018 óta a MediaKind vezető videótömörítési mérnöke, elsősorban az AI és ML alkalmazásával foglalkozik, médiafeldolgozó és -továbbítási eljárásokra összpontosítva. Ezek az eljárások a kódoló optimalizálástól az olyan új területekig terjednek, mint a szuperfelbontás és a videószemantikai elemzés.

BORDÁS CSABA 1993-ban szerzett mérnöki oklevelet a Kolozsvári Műszaki Egyetem gyengeáramú és távközlési szakán, majd 1993 és 2001 között Marosvásárhelyen dolgozott elsősorban az erdélyi analóg telefonhálózat digitalizációjában. Közben asszisztensként tevékenykedett az induló Gábor Dénes Főiskola helyi tagozatában. 2001-től csatlakozott a Ericsson csapatához Budapesten, először rendszertervezőként, majd a vezetékes ügyfélkör és a Magyar Telekom értékesítési ágazata műszaki igazgatójaként. Főleg FTTx-hozzáféréssel és IP-hálózati megoldásokkal foglalkozott, részt vett a hazai ADSL-, VDSL- és FTTH- hálózatok kiépítésében. 2015-től az Ericsson Deutsche Telekom globális értékesítési ágazat TV&Media igazgatója, ahol elsősorban a DT leányvállalatainak TV-megoldásaival, ezek fejlesztéseivel foglalkozott. 2018 óta hasonló szerepben a Deutsche Telekom ügyfélkör TV- és média-üzletágát koordinálja a MediaKind-nál.