



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Gépészmérnöki Kar

Energetikai Gépek és Rendszerek Tanszék

Illés Dávid Károly

**ERŐMŰ VISELKEDÉSÉNEK LEÍRÁSA
NEURÁLIS HÁLÓ SEGÍTSÉGÉVEL**

KONZULENS

Dr. Groniewsky Axel

BUDAPEST, 2022

Lezárás dátuma: 2022. 11. 16.

Tartalomjegyzék

1. Bevezetés és célkitűzések	2
2. A neurális hálók alapjai	4
2.1. Mesterséges neurális hálók biológiai háttere	4
2.2. Mesterséges neurális háló szerkezeti és matematikai áttekintése	5
2.2.1. A rétegek kimeneteinek számítása	5
2.2.2. A paraméterek változásának meghatározása	8
2.2.3. Paraméterek frissítése.....	9
3. Modern neurális háló architektúrák	11
3.1. A rekurrens neurális háló (RNN)	11
3.2. A konvolúciós neurális háló (CNN).....	12
4. Neurális háló alkalmazása erőművi rendszerek leírására.....	14
4.1. Teljesen kapcsolt neurális hálók alkalmazása	14
4.2. Rekurrens neurális hálók alkalmazása	14
4.3. Konvolúciós neurális hálók alkalmazása	15
5. A termodinamikai modellre alapozott neurális háló elkészítése	16
5.1. A CycleTempo modell.....	16
5.2. A háló tanításához szükséges adathalmaz létrehozása.....	18
5.3. Az adatok előkészítése	21
5.4. Különböző hálóarchitektúrák vizsgálata.....	21
5.4.1. Egy rejtett réteget tartalmazó architektúrák	22
5.4.2. Két rejtett réteget tartalmazó architektúrák	28
5.4.3. Tapasztalatok összefoglalása	34
5.5. A tanító és teszt adatsor kiegyensúlyozása.....	35
5.6. A vizsgált architektúrák viselkedése a kiegyensúlyozott adatsoron	36
5.6.1. Egy rejtett rétegű architektúrák viselkedése a kiegyensúlyozott adatsoron	36
5.6.2. Két rejtett rétegű architektúrák viselkedése a kiegyensúlyozott adatsoron	37
5.7. A bemeneti változók csökkentése.....	38
6. Összefoglalás	44
7. Felhasznált források	46
8. Melléklet.....	48

1. BEVEZETÉS ÉS CÉLKITŰZÉSEK

Egy pontos, minél több részletre kiterjedő komplex termodinamikai modell elkészítése különösen fontos, kiváltképp energiatermelő létesítmények tervezése vagy átalakítása esetén. Az ilyen típusú beruházások gazdasági megtérülése nem prognosztizálható az üzemvitel, valamint a változó költségek ismerete nélkül. Az iparban elterjedt modellező szoftverek, mint az 1981-ben megjelent, ma a GE által fejlesztett GateCycle™, illetve a STEAG Energy Services által közel 30 éve finomhangolt EBSILON® Professional egyaránt alkalmasak egy erőmű névleges és részterhelésen működő üzemállapotainak előrejelzésére, továbbá a leadott villamos- és hőteljesítmények, illetve hatásfokok számítására figyelembe véve például a környezeti paramétereket, valamint a berendezések degradációját. [1] [2]

Az előző bekezdésben említett szoftverek a modellezett rendszerre felírt tömeg- és energiamérlegek, sztöchiometriai, illetve termodinamikai összefüggések alapján határozzák meg az üzemi paramétereket. A rendszer különböző munkapontjait leíró tömegáramok és állapotjelzők számítása iteratív eljárással történik, amely komoly kapacitásokat képes lekötni hardver oldalon, így a kalkuláció időigényessé válhat, főképp, ha összetett technológiáról van szó.

A fentebb említett problémakör megoldása érdekében az ipar innovatívabb optimalizációs, illetve regresszió vizsgálati eljárások felé fordult. Hőkörfolyamati modellezésben egyre gyakoribb a genetikus algoritmusok, nemlineáris programozás és neurális hálók alkalmazása [3]. Jelen dolgozat az utóbbi, mesterséges intelligencián alapuló diszciplínára fókuszál.

A neurális hálók, illetve általában a gépi tanulási módszerek nagy mennyiségű bemenő adatot igényelnek. Innen is látszik, hogy a módszer kitűnően alkalmas energiatermelő létesítmények modellezésére, hiszen azokban rengeteg információ keletkezik nagy felbontásban a digitális mérőműszereknek hála. A neurális hálók hőséma számításba történő implementálásával komoly eredményeket értek el a gyakorlatban. A következőkben ezen eredmények és alkalmazások közül néhányat fogok bemutatni.

Dolgozatomban egy olyan neurális háló elkészítését tűztem ki célul, amely egy CycleTempo modellező szoftverben elkészített szuperkritikus szénerőmű nettó hatásfokát képes prognosztizálni. A háló architektúra optimalizálása során különböző számú rejtett rétegekkel, neuronokkal, eltérő aktivációs függvényekkel, illetve az adatsor bizonyos transzformációival kísérleteztem. Megvizsgáltam, hogy a választott magyarázó változók különböző ismétlés nélküli kombinációival milyen pontosságú becslést szolgáltat az optimalizált háló. Ezzel meghatároztam a hálót leíró üzemi paraméterek számától függő legjobb bemeneti paraméterkombinációkat. Eredményeim általános érvényű útmutatást nyújtanak egy hőerőmű nettó hatásfokának neurális hálóval való minél pontosabb leírásához.

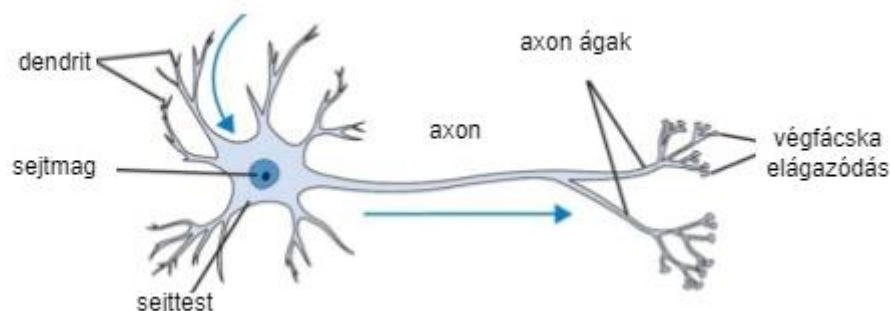
A konkrét kutatási eredmények taglalása előtt, azonban megismertetem az Olvasót a neurális hálók alapvető működésével és matematikai hátterével, valamint megvizsgálom a legmodernebb neurális háló architektúrák alkalmazási lehetőségeit, illetve jellemzőit.

2. A NEURÁLIS HÁLÓK ALAPJAI

E fejezetben a neurális hálók regresszió vizsgálatban történő alkalmazását, valamint az ezen problémakör kezeléséhez kapcsolódó strukturális és matematikai sajátosságokat fogom bemutatni egy egyszerű teljesen kapcsolt („fully connected”) architektúrájú modellen keresztül. Elsőként azonban a mesterséges neurális hálók biológiai asszociációs hátterét fogom kifejezni röviden.

2.1. Mesterséges neurális hálók biológiai háttere

Az emberi agy körülbelül 10^{11} számú neuront tartalmaz, melyek szinapszisokon keresztül kapcsolódnak egymáshoz, kémiai vagy elektromos kölcsönhatás által. A neuronba az ún. dendriten fut be az ingerület, vagyis a szétterjedő potenciálváltozás. A dendrit az ingerületet a sejttest felé továbbítja, onnan pedig az axonon keresztül jut el a környező neuronok felé [12]. Ha a dendriten kialakult gerjesztés elér egy bizonyos küszöbértéket, akkor az információ továbbítódik az axonon, utána viszont a neuron inaktívvá válik, tehát az információáramlás szakaszos jellegű. Az, hogy az ingerület mely ágon továbbítódik, az a helyileg kialakult potenciálviszonyoktól függ. A 2-1. ábra egy neuron felépítését mutatja be.



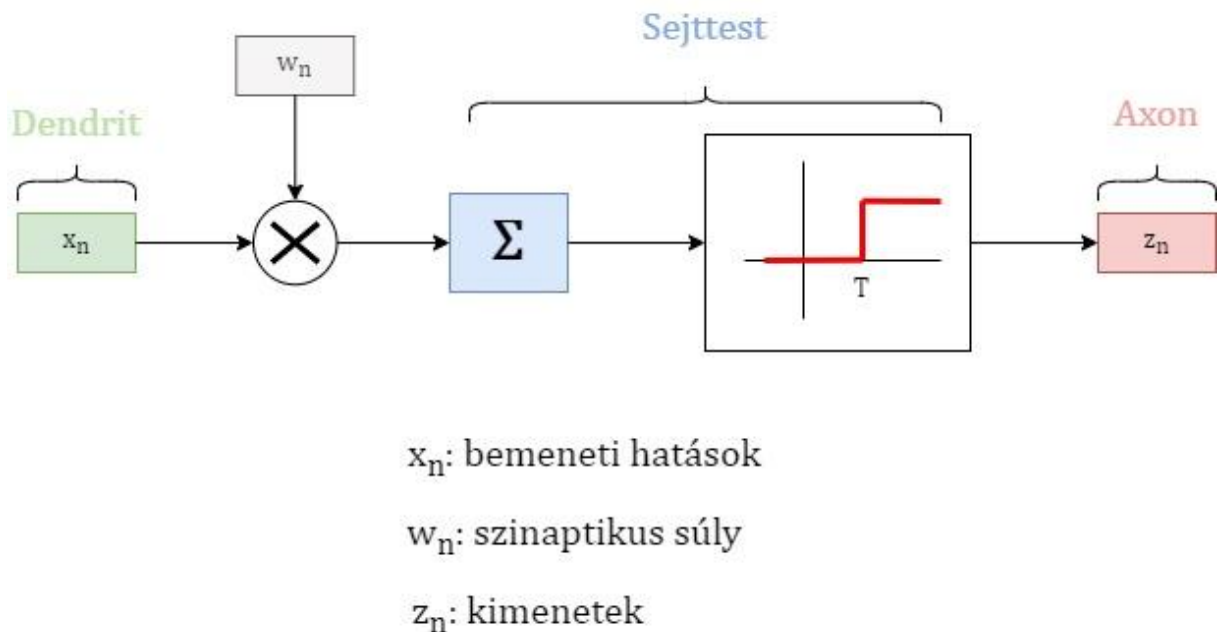
2-1. ábra: Neuron felépítése [12]

A biológiai neuronok matematikai reprezentációjakor csupán a következő mechanizmusokat vették figyelembe:

- A neuronok vagy továbbítják az információt vagy nem, azaz binárisan modellezendők.
- A bemenetek hatásai összeadódnak.
- Az erősebb kapcsolat nagyobb potenciálkülönbséggel jön létre, ezt súlyozással érdemes a modellben megjeleníteni.

Összefoglalva tehát a bemenetek kezeléséért a dendrit felelős. Ha a sejttestben a súlyozott hatások összegzéséből adódó gerjesztés elér egy T küszöbértéket, akkor a neuron aktiválódik vagyis T -nél aktivációs értéke 0-ról egy bizonyos értékre vált és a ki-menet megjelenik az axonon, melyen keresztül egy kapcsolódó neuronhoz továbbítódik az információ. Az előbbieken leírt neuron aktivációs függvény nem folytonos,

egységugrás jellegű függvény, amelyről fontos megjegyezni, hogy nem deriválható. E ténynek kiemelt jelentősége lesz a 2.2.1 fejezetre nézve. A 2-2. ábra a biológiai neuron működéséből származtatott matematikai modellt ábrázolja [13].



2-2. ábra: A biológiai neuron matematikai modellje

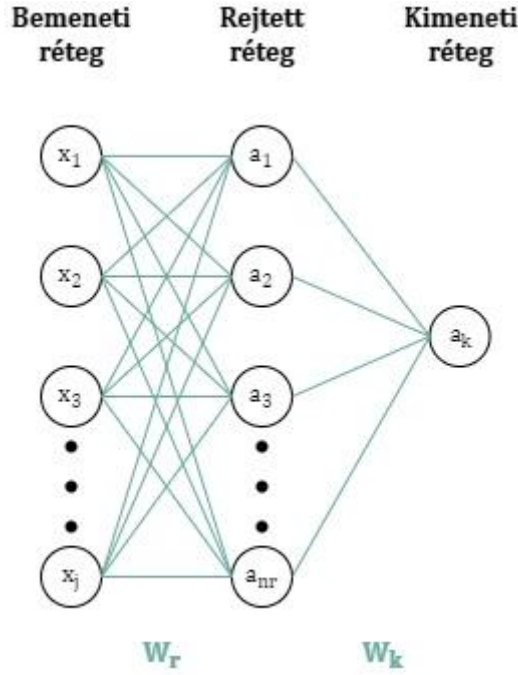
A mesterséges neurális hálókat a fenti modellre alapozva alkották meg, majd egyes részeit módosítva fejlesztették azt.

2.2. Mesterséges neurális háló szerkezeti és matematikai áttekintése

A hálót egyelőre fekete doboz modellként kezelve bemenetként a vizsgált adathalmazhoz tartozó magyarázó változók értékeit, míg kimenetként a magyarázott változó értékét tekintjük. A felépített háló segítségével a modell bemenetei és kimenete között fennálló összefüggésrendszer meghatározása a cél. A modellparaméterek változtatása iteratív eljárás során történik, melyet a háló tanulási folyamatának nevezünk. Egy iteráció a kimenetek számításából, a paraméterek szükséges változtatásának meghatározásából, majd ezen változtatások szerinti paraméterfrissítésből áll. A modell általánosító, prognosztizáló erővel rendelkezik, amennyiben a produkált kimenetek megfelelő mértékben illeszkednek a várt értékekhez.

2.2.1. A RÉTEGEK KIMENETEINEK SZÁMÍTÁSA

A következőkben tárgyalunk teljesen kapcsolt neurális háló vertikálisan három részre, azaz 1 db bemeneti, 1 db rejtett és 1 db kimeneti rétegre tagolható. A vizsgált háló struktúráját az 2-3. ábra szemlélteti.



2-3. ábra: Teljesen kapcsolt neurális háló architektúra regressziós probléma megoldására

A bemeneti réteg a magyarázó változók értékeit tartalmazza, viszont ez nem tekinthető a háló részének, mivel nem végez számítást, nem tartalmaz modellparamétert. A bemeneti réteg matematikailag egy $(m \times j)$ dimenziójú mátrixszal reprezentálható ($\mathbf{X}$), ahol m a minták vagy mérési pontok, j a magyarázó változók száma.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} \end{bmatrix} = \mathbf{X}$$

A rejtett réteg ún. neuronokat foglal magában, melyek értékei mátrixműveletek és az alkalmazott aktivációs függvény szerint állnak elő. A neuronok és a bemeneti réteg közötti kapcsolatot egy $(n_r \times j)$ dimenziójú $\mathbf{W}_r$ súlymátrix, illetve egy $\bar{b}_r$ eltolásvektor $(n_r \times 1)$ teremti meg, ahol n_r a rejtett rétegbeli neuronok száma. A rejtett réteghez tartozó súlymátrixot és eltolásvektort általában random $[0;1]$ értékekkel inicializáljuk.

$$\begin{bmatrix} w_{11}^{(r)} & w_{12}^{(r)} & \cdots & w_{1j}^{(r)} \\ w_{21}^{(r)} & w_{22}^{(r)} & \cdots & w_{2j}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_r 1}^{(r)} & w_{n_r 2}^{(r)} & \cdots & w_{n_r j}^{(r)} \end{bmatrix} = \mathbf{W}_r \quad \begin{bmatrix} b_1^{(r)} \\ b_2^{(r)} \\ \vdots \\ b_{n_r}^{(r)} \end{bmatrix} = \bar{b}_r$$

A rejtett réteget egy $(n_r \times m)$ dimenziójú mátrix képviseli ($\mathbf{Z}_r$), mely a súlymátrix és a transzponált bemeneti mátrix elemeinek - a mátrix szorzás szabályainak megfelelő - lineáris kombinációjaként, valamint az eltolásvektorral korrigálva jön létre (1).

$$\begin{aligned}
& \begin{bmatrix} w_{11}^{(r)} & w_{12}^{(r)} & \cdots & w_{1j}^{(r)} \\ w_{21}^{(r)} & w_{22}^{(r)} & \cdots & w_{2j}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_r1}^{(r)} & w_{n_r2}^{(r)} & \cdots & w_{n_rj}^{(r)} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{m1} \\ x_{12} & x_{22} & \cdots & x_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1j} & x_{2j} & \cdots & x_{mj} \end{bmatrix} + \begin{bmatrix} b_1^{(r)} \\ b_2^{(r)} \\ \vdots \\ b_{n_r}^{(r)} \end{bmatrix} = \begin{bmatrix} z_{11}^{(r)} & z_{12}^{(r)} & \cdots & z_{1m}^{(r)} \\ z_{21}^{(r)} & z_{22}^{(r)} & \cdots & z_{2m}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n_r1}^{(r)} & z_{n_r2}^{(r)} & \cdots & z_{n_rm}^{(r)} \end{bmatrix} \\
& \quad \quad \quad (n_r \times j) \quad \quad \quad (j \times m) \quad \quad \quad (n_r \times 1) \quad \quad \quad (n_r \times m) \\
& \quad \quad \quad \mathbf{W}_r \mathbf{X}^T + \bar{\mathbf{b}}_r = \mathbf{Z}_r \quad \quad \quad (1)
\end{aligned}$$

A rejtett réteg eddig a pontig lényegében egy, a megszokottnál komplexebb többváltozós lineáris regressziót valósít meg. Ahhoz, hogy a modell a vizsgált adathalmazra egy bonyolultabb összefüggés szerinti illesztést is el tudjon végezni, a rejtett réteg mátrixát egy nemlineáris aktivációs függvény argumentumába helyezzük. Ennek a modellparaméterek, vagyis a súlymátrix és eltolásvektor változtatásának meghatározása során kiemelt jelentősége lesz (lásd 2.2.2. fejezet). A 2.1 fejezetben leírt egységugrás jellegű, biológiai analógiából ismert aktivációs függvény helyett a differenciálhatóság érdekében valamilyen folytonos függvényt alkalmazunk. Így matematikailag könnyen értelmezhetővé válnak a háló tanulásához szükséges számítások, illetve a kimenetek értékei bináris skála helyett folytonos skálán jelennek meg, azaz finomabb felbontású adatokkal megy végbe a kalkuláció.

Ilyen függvény ($f(x)$) lehet a *ReLU* (*rectified linear unit*) vagy bármilyen szigmoid jellegű formula, mely $[0;x]$ vagy $[-1;1]$ értékészlettel újraskálázza $\mathbf{Z}_r$ mátrix elemeit, így $\mathbf{A}_r$ lesz a rejtett réteg kimenete a (2) összefüggés szerint. Fontos megjegyezni, hogy elsősorban az aktivációs függvény alakja miatt a modellben szereplő összes bemeneti és modellparaméter értékét egy szűk, nagyjából $[-2, 2]$ tartományban kell tartani a számítások során, hiszen így a lineáris kombinációkban szereplő tagok közül egyik sem lesz domináns. Rétegenként eltérő vagy megegyező aktivációs függvényeket is alkalmazhatunk a bemenet és kimenet között fennálló összefüggésrendszer jellegétől, komplexitásától függően.

$$f_r(\mathbf{Z}_r) = \mathbf{A}_r \quad (2)$$

A kimeneti réteg regressziós probléma esetén 1 db neuronból áll. Ennek a háló paraméterváltoztatási mechanizmusához van köze és a 2.2. fejezetben fejtem ki bővebben. A kimeneti réteghez tartozó $\mathbf{W}_k$ súlymátrix ($1 \times n_r$), $\bar{\mathbf{b}}_k$ eltolásvektora (1×1) dimenziójú. Jelen réteg bemeneti mátrixát a rejtett réteg kimenete ($\mathbf{A}_r$) képezi. A kimeneti réteg neuronjának, azaz a regressziós modell predikciójának értéke ($\mathbf{A}_k$) a (3) összefüggés szerint határozható meg.

$$\begin{aligned}
& f_k \left(\begin{bmatrix} w_{11}^{(k)} & w_{12}^{(k)} & \cdots & w_{1n_r}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_r1}^{(k)} & w_{n_r2}^{(k)} & \cdots & w_{n_rn_r}^{(k)} \end{bmatrix} \begin{bmatrix} a_{11}^{(r)} & a_{12}^{(r)} & \cdots & a_{1m}^{(r)} \\ a_{21}^{(r)} & a_{22}^{(r)} & \cdots & a_{2m}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_r1}^{(r)} & a_{n_r2}^{(r)} & \cdots & a_{n_rm}^{(r)} \end{bmatrix} + \begin{bmatrix} b_1^{(k)} \end{bmatrix} \right) \\
& \quad \quad \quad (1 \times n_r) \quad \quad \quad (n_r \times m) \quad \quad \quad (1 \times 1) \\
& \quad \quad \quad = f_k \left(\begin{bmatrix} z_{11}^{(k)} & z_{12}^{(k)} & \cdots & z_{1m}^{(k)} \end{bmatrix} \right) \\
& \quad \quad \quad (1 \times m)
\end{aligned}$$

$$f_k(\mathbf{W}_k \mathbf{A}_r + \bar{b}_k) = \bar{A}_k \quad (3)$$

2.2.2. A PARAMÉTEREK VÁLTOZÁSÁNAK MEGHATÁROZÁSA

Neurális hálók esetén a kimeneti réteg meghatározása után egy hibafüggvény definiálása szükséges, amely az adott iterációban végzett predikció pontosságát írja le a várt magyarázott változó ($\bar{Y}$) értékéhez képest. Regresszióvizsgálat esetén az átlagos négyzetes hibaösszeg (MSE - mean squared error) használata a legkézenfekvőbb [20]:

$$MSE = \frac{1}{m} \sum_1^m (\bar{A}_k - \bar{Y})^2. \quad (4)$$

A mérési pontok átlagos négyzetes hibáit az $(1 \times m)$ dimenziójú C vektor hordozza [20]:

$$C = \frac{1}{m} (\bar{A}_k - \bar{Y})^2. \quad (5)$$

A modell annál pontosabb illesztést végez a vizsgált adathalmazra, minél kisebb a MSE értéke. Célunk tehát az, hogy iterációról iterációra csökkentjük a MSE értékét a súlymátrixok és eltolásvektorok megfelelő változtatásán keresztül. Ahhoz, hogy meghatározzuk a hibafüggvény súly és eltolás értékektől való függését, parciális deriválást és a láncszabályt kell alkalmaznunk. A kimeneti réteg paraméterváltozásának számítását a következő összefüggések írják le:

$$\frac{\partial C}{\partial W_k} = \frac{\partial C}{\partial A_k} \frac{\partial A_k}{\partial Z_k} \frac{\partial Z_k}{\partial W_k} \quad (6)$$

$$\frac{\partial C}{\partial A_k} = \frac{2}{m} (\bar{A}_k - \bar{Y}) \quad (7)$$

$$\frac{\partial A_k}{\partial Z_k} = f'_k(\bar{Z}_k) \quad (8)$$

$$\frac{\partial Z_k}{\partial W_k} = \mathbf{A}_r \quad (9)$$

$$\frac{\partial C}{\partial W_k} = \frac{2}{m} (\bar{A}_k - \bar{Y}) \circ \frac{f'_k(\bar{Z}_k)}{(1 \times m)} \cdot \frac{\mathbf{A}_r^T}{(m \times \mathbf{n}_r)} = \frac{dW_k}{(1 \times \mathbf{n}_r)} \quad (10)$$

$$\frac{\partial C}{\partial b_k} = \frac{\partial C}{\partial A_k} \frac{\partial A_k}{\partial Z_k} \frac{\partial Z_k}{\partial b_k} \quad (11)$$

$$\frac{\partial Z_k}{\partial b_k} = 1 \quad (12)$$

$$\frac{\partial C}{\partial b_k} = \sum_1^m \left(\frac{2}{m} (\bar{A}_k - \bar{Y}) \circ f'_k(\bar{Z}_k) \right) = \frac{db_k}{(1 \times 1)} \quad (13)$$

A rejtett réteg paraméterváltozásának meghatározásához a fentiekkel analóg összefüggések használhatóak:

$$\frac{\partial C}{\partial W_r} = \frac{\partial C}{\partial A_r} \frac{\partial A_r}{\partial Z_r} \frac{\partial Z_r}{\partial W_r} \quad (14)$$

$$\frac{\partial C}{\partial A_r} = \frac{\partial C}{\partial A_k} \frac{\partial A_k}{\partial Z_k} \frac{\partial Z_k}{\partial A_r} \quad (15)$$

$$\frac{\partial Z_k}{\partial A_r} = \mathbf{W}_k \quad (16)$$

$$\frac{\partial C}{\partial A_r} = \mathbf{W}_k^T \cdot \frac{2}{m} (\overline{A_k} - \bar{Y}) \circ f'_k(\overline{Z_k}) = \mathbf{dA}_r \quad (17)$$

$(n_r \times 1) \quad (1 \times m) \quad (1 \times m) \quad (n_r \times m)$

$$\frac{\partial A_r}{\partial Z_r} = f'_r(\overline{Z_r}) \quad (18)$$

$$\frac{\partial Z_r}{\partial W_r} = \mathbf{X}^T \quad (19)$$

$$\frac{\partial C}{\partial W_r} = f'_r(\overline{Z_r}) \circ \mathbf{dA}_r \cdot \mathbf{X} = \mathbf{dW}_r \quad (20)$$

$(n_r \times m) \quad (n_r \times m) \quad (m \times j) \quad (n_r \times j)$

$$\frac{\partial C}{\partial b_r} = \frac{\partial C}{\partial A_r} \frac{\partial A_r}{\partial Z_r} \frac{\partial Z_r}{\partial b_r} \quad (21)$$

$$\frac{\partial Z_r}{\partial b_r} = 1 \quad (22)$$

$$\frac{\partial C}{\partial b_r} = \sum_1^m (dA_r \circ f'_r(\overline{Z_r})) = \overline{db_r} \quad (23)$$

$(n_r \times 1)$

Amennyiben a vizsgált probléma több magyarázott változóra is kiterjed, vagyis a kimeneti réteg 1 db neuronnál többet tartalmaz, akkor minden egyes neuronhoz külön hibafüggvényt kell definiálni. A konvergenciát regresszióanalízis esetén azonban gátolja, hogy a hibafüggvények A_r szerinti parciális deriváltjai más-más értéket vehetnek fel, mivel ugyanazon rejtett rétegbeli neuronhoz több kimeneti neuron is kapcsolódik. Így a számított dA_r a kimeneti neuronokhoz tartozó parciális deriváltak összege lesz, vagyis a hibafüggvények összege fog konvergálni egy minimumhoz és nem pedig az egyes predikciók lesznek egyre pontosabbak. Máshogy fogalmazva a háló minden magyarázott változóra adott becslése „ugyanolyan pontatlan” lesz. Az előzőek alapján már belátható, hogy regressziós feladat esetén miért használunk 1 db neuront a kimeneti rétegben.

2.2.3. PARAMÉTEREK FRISSÍTÉSE

Miután meghatároztuk a hibafüggvény csökkentését szolgáló paraméterváltozásokat, frissítenünk kell az egyes rétegekhez tartozó súlymátrixot, illetve eltolásvektort, hogy a következő iterációban már ezekkel számolja újra a modell saját kimenetét. Ehhez bevezetésre kerül az ún. tanulási ráta (α), mely értéke a konvergencia gyorsaságát befolyásolja. Túl alacsony α esetén több iteráció lefutását igényli a modell, míg nagy

értékű α esetén annak a veszélye áll fent, hogy a hibafüggvény valódi minimuma helyett csak egy lokális minimumot találunk meg. A tanulási ráta értékét nagyjából 0,1 körüli értéken érdemes felvenni. A paraméterek frissítése az alábbi egyszerű összefüggések szerint történik, ahol i egy adott iterációs ciklus indexe:

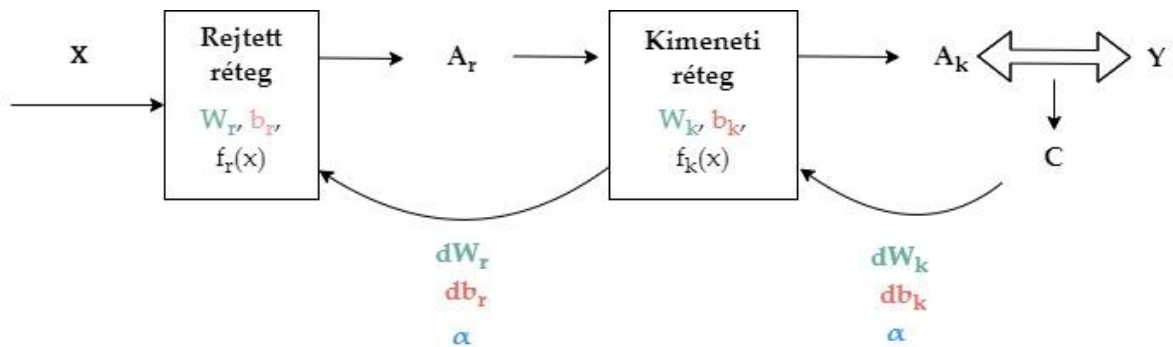
$$W_{k_{i+1}} = W_{k_i} - \alpha dW_{k_i} \quad (24)$$

$$\overline{b_{k_{i+1}}} = \overline{b_{k_i}} - \alpha d\overline{b_{k_i}} \quad (25)$$

$$W_{r_{i+1}} = W_{r_i} - \alpha dW_{r_i} \quad (26)$$

$$\overline{b_{r_{i+1}}} = \overline{b_{r_i}} - \alpha d\overline{b_{r_i}} \quad (27)$$

Összefoglalva tehát az iteráció első lépéseként random értékeken inicializáljuk a súlymátrixokat és eltolásvektorokat, majd kiszámítjuk a kimenetet, vagyis a modell által nyújtott predikciót. Ezt összevetjük a várt értékekkel, így képezünk egy hibafüggvényt, és olyan modellparaméter-változásokat határozzunk meg, melyek csökkentő hatással lesznek a hibafüggvényre a következő iterációban. Minden egyes ciklust a modellparaméterek frissítése zárja, ezek után kezdődhet a kimenet újraszámítása, ezúttal már a frissített súlymátrixok, illetve eltolásvektorok segítségével. Az előbbieken leírt eljárást addig ismételjük, míg a modell által nyújtott kimenet megfelelően nem illeszkedik a várt értékekhez. A tanulási folyamat szemléltetését a 2-4. ábra mutatja be.



2-4. ábra: Teljesen kapcsolt neurális háló tanulási folyamata

A rejtett rétegek száma, az abban megtalálható neuronok száma, valamint a tanulási ráta nem változik a tanulási folyamat során. Ezeket a háló hiperparamétereinek nevezzük. A rejtett rétegek, valamint az abban foglalt neuronok számát a probléma komplexitásának megfelelően célszerű növelni.

3. MODERN NEURÁLIS HÁLÓ ARCHITEKTÚRÁK

A hagyományos, teljesen kapcsolt architektúrájú neurális hálók egy erőmű állandósult állapotainak matematikai viselkedését képesek megfelelően modellezni, mert az input adatok sorrendisége irreleváns a számítások elvégzésekor, illetve a probléma megoldásához szükséges paraméterszám sem túl magas.

Modernebb hálótípust igényelhet olyan erőművi rendszer vizsgálata, mely esetén az adott pillanatbeli vagy jövőbeli állapotot csak a múltbeli állapotok ismeretén keresztül lehet meghatározni. Ebből következik, hogy ilyenkor a bemenő adatok sorrendjének is kiemelt fontossága van. Tehát berendezés- vagy rendszerszintű tranziens üzemállapotok modellezésekor ún. rekurrens neurális háló alkalmazása lehet indokolt.

Amennyiben a bemenő paraméterek sokasága válik a teljesen kapcsolt háló számára kezelhetetlenné, akkor konvolúciós rétegekkel érdemes az adathalmaz méretét csökkenteni. Erőművi területen ennek tipikusan valamilyen osztályozási feladatra visszavezethető diagnosztikai problémánál lehet szerepe. A rekurrens és konvolúciós hálók főbb tulajdonságait a következő alfejezetekben foglaltam össze.

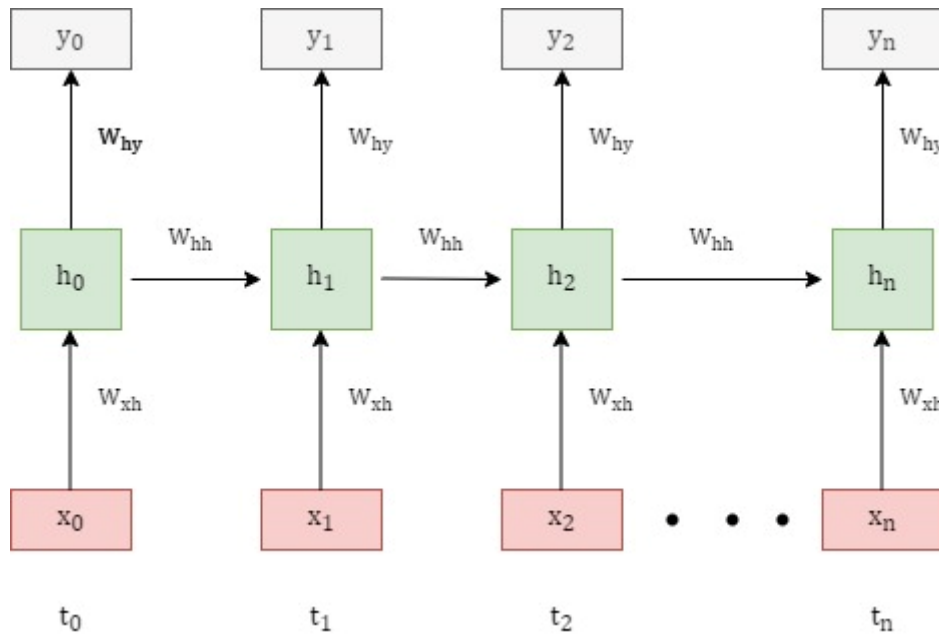
3.1. A rekurrens neurális háló (RNN)

Rekurrens neurális hálót elsősorban hisztorikus adatok feldolgozásakor szokás alkalmazni. A leglényegesebb különbség, amiben eltér a teljesen kapcsolt hálótól hogy az RNN rejtett rétegeiben lévő neuronok visszacsatolással rendelkeznek, melyek biztosítják az adatok szekvenciális jellegének felismerését [9]. Máshogy fogalmazva, a rekurrens réteg kimenete saját bemeneteként is szolgál a valamilyen Δt idővel későbbi állapot kiszámításakor [10].

Az input (x_t) és output ($\hat{y}_t$) vektort minden t időlépés esetén egy ún. belső állapot (h_t) köt össze, mely nem csak a bemenet vektorától, de az előző állapottól is függ. Az algoritmus végig iterál az összes időlépésen, és ciklusról-ciklusra frissíti a belső állapotokat, illetve számolja a kimeneteket. További érdekes aspektusa ennek a háló típusnak, hogy ún. paraméter megosztási tulajdonsággal rendelkezik, ami azt jelenti, hogy a bemenet-belső állapot (W_{xh}), belső állapot-kimenet (W_{hy}), valamint a két állapot összeköttetését jelentő súlymátrix (W_{hh}) minden egyes időlépésre megegyezik [9]. Az RNN-hez kapcsolódó matematikai összefüggéseket a (28-29) képletek tartalmazzák, a számítás időbeli lefutását a 3-1. ábra szemlélteti [15].

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t) \quad (28)$$

$$\hat{y}_t = W_{hy}^T h_t \quad (29)$$



3-1. ábra: RNN számítási lefutása az időben

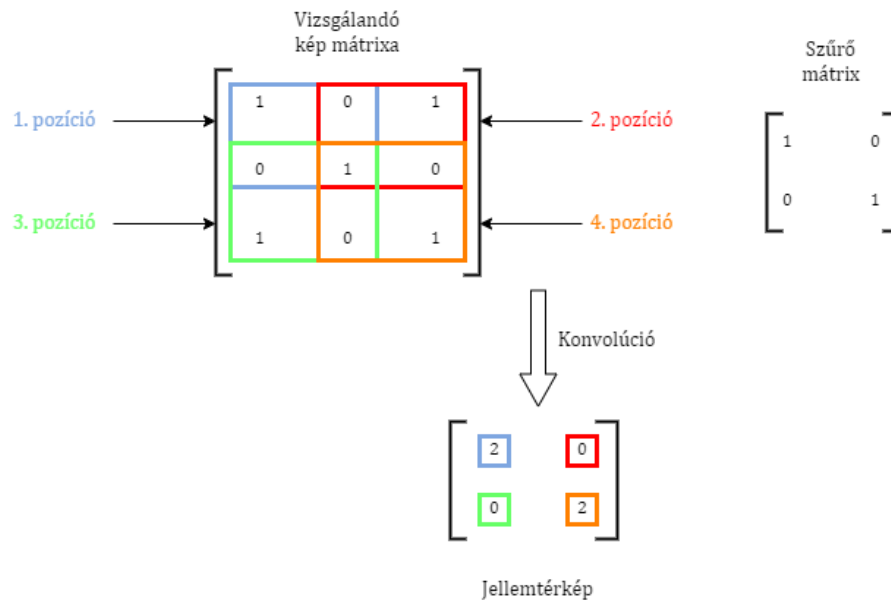
Problémát jelent RNN-ek esetében, hogy az időlépcső kalkulációja információvesztéssel jár, vagyis néhány lépés után a háló már nem hordozza a bemenő adatok tulajdonságait. Ennek feloldásához olyan rétegtípusokkal kell kombinálni az RNN-t, amelyek képesek eldönteni, hogy mely kimenetek legyenek tárolva hosszútávon, mely tárolt objektumok legyenek beolvasva a következő időlépcső meghatározásakor vagy éppen mely tárolt adat vált feleslegessé az adott iterációban. Ilyen a „long-short term memory” vagy a „gated recurrent unit” rétegtípus. Nehézségként jelentkezhethet, hogy a hibafüggvény rekurzív differenciálása miatt nagy mértékben megnőhet a háló tanítási ideje, azonban általában ezt az LSTM és GRU architektúrák szintén képesek kezelni, mivel olyan elágazásokat hoznak létre, melyek deriváltjai nem hatalmasodnak vagy tűnnek el [10].

3.2. A konvolúciós neurális háló (CNN)

A konvolúciós hálók (CNN) tipikusan nagy számú bemeneti paraméter kezelésére, így képek vizsgálatára alkalmasak. A kép általános jellemzőinek felismerésén keresztül lehet azt bizonyos kategóriákba sorolni, vagyis osztályozni. Hagyományos teljesen kapcsolt neurális hálókkal a hasonló problémákat nehéz kezelni, mivel a 2D kép mátrixát vektorra alakítva, annak térbeliségét teljesen leromboljuk, ráadásul a nagy számú bemenő paraméter olyan komplexsége teszi a hálót, hogy a számítások túlságosan lelassulnak. A cél tehát az, hogy a bemenő adatok dimenzió számát úgy csökkentsük, hogy a pixelek térbeli elhelyezkedéséről szóló információkat részben megőrizzük. A konvolúciós háló az előbb említett műveletet végzi el az input adatsoron.

A háló először egy konvolúciót valósít meg ún. szűrők segítségével, melyeket a tanulási folyamat során frissülő súlymátrixok reprezentálnak [9]. A súlymátrixok felvételéhez ismerni kell a keresett mintákat. A szűrő dimenziója mindenképpen kisebb,

mint a vizsgált képé. A súlymátrix-szal „letapogatjuk” a képet, vagyis Hadamard-szorzatokat¹ képezünk, a kapott minor értékeit pedig összeadjuk. Az összeg a szűrő alkalmazásának pozíciójában rögzül. A keresett minta azon pozíciókban jelenik meg, ahol ez az összeg nagyobb. A képzett kisebb dimenziójú mátrixokat „jellemtéreképeknek” nevezzük. A konvolúció szemléltetését a 3-2. ábra mutatja be [16]. A példában egy átlós mintát kerestünk a vizsgálandó képen. A keresett jellemzőt az adott 2x2-es minorban képzett Hadamard-szorzat után a jellemtérkép megfelelően tükrözi.



3-2. ábra: Konvolúció szemléltetése

A jellemtéreképek értékei ezután a 2.2.1 fejezetben megismert valamely nemlineáris aktivációs függvény argumentumába kerülnek, majd a „pooling” réteg tovább csökkenti a dimenziószámot. Gyakran alkalmazott módszer „pooling” rétegek esetén, hogy a jellemtérképet annak legnagyobb értékével reprezentáljuk a továbbiakban. A konvolúciós, nemlineáris és „pooling” rétegeket akár többszörözve is alkalmazhatjuk a hálóban. Az input adatok paraméterszámának csökkentése után az osztályozási feladatot már egy szimpla teljesen kapcsolt háló végzi el [16].

¹ A Hadamard-szorzat, más néven Schur-szorzat vagy elemenkénti szorzat egy kétváltozós művelet, aminek tényezői azonos dimenziójú mátrixok. A szorzatban álló elemek a tényezők megfelelő elemeinek szorzatai, így a szorzat dimenziója is megegyezik a tényezők dimenziójával.

4. NEURÁLIS HÁLÓ ALKALMAZÁSA ERŐMŰVI RENDSZEREK LEÍRÁSÁRA

4.1. Teljesen kapcsolt neurális hálók alkalmazása

Olyan szofisztikált termodinamikai rendszer esetén, mint egy 50 MW_e villamos teljesítményű, szuperkritikus Brayton – ORC kombinált ciklus, sikerült elérni közel 60%-os számítási időmegtakarítást annak köszönhetően, hogy a kazánban, illetve a hőcserélőkben létrejött nyomásesés becslését neurális háló segítségével végezték [3]. Egy másik kutatásban kapcsoltan termelő létesítmény poszt-retrofit energiafogyasztását, ennek segítségével pedig a beavatkozás után adódó energiamegtakarítást számították ki nagy pontossággal [4]. A svéd Helsingborgban található szintén kapcsoltan termelő fűtőerőműben pedig valós idejű monitoring rendszert alakítottak ki neurális háló alapon. Az alkalmazott modell érdekessége, hogy a hőforrás és turbina alrendszer külön hálóval lett reprezentálva, így az előbbi háló kimeneti paraméterei szolgáltak utóbbi egyes bemeneteiként, ezzel egyesítve az alrendszereket [5].

A szakirodalmi források alapján általánosan kijelenthető, hogy a neurális hálók erőművi rendszerek modellezésében történő használata különösen akkor indokolt és hatékony, ha

- az adott technológia összetettsége miatt az egyes rendszerek együttes optimalizációja túl nagy számítási kapacitást köt le [3],
- az adott létesítmény már üzemel, de nem létezik rá fizikai modell [5],
- az adott létesítmény valós idejű szimulációja kívánatos [4],
- az adott rendszert leíró analitikus összefüggés nem ismert.

A neurális hálók előnye, hogy az erőművi folyamatokban fellelhető nemlineáris összefüggéseket képesek lekövetni. Hátrányuk azonban, hogy a tanításon kívüli paramétertartományokban nem szolgáltatnak pontos prognózist, valamint, hogy a modellezett rendszer bármilyen módosulása esetén a hálót újra kell tanítani. Nagyon fontos megjegyezni, hogy a modellezendő technológia átfogó ismerete elengedhetetlen, főként a bemenő paraméterek megválasztásakor [4].

4.2. Rekurrens neurális hálók alkalmazása

A rekurrens neurális hálók erőműiparban történő alkalmazására kiváló példa, hogy GRU RNN kombinált hálózatot használva sikeresen jelezték előre egy szén tüzelésű erőmű újrahevítő felületeinek hőmérsékleteit idősoros adatokra alapozva. A modell 5 perces jövőbeli prognózist tudott nyújtani, mely már lehetővé teheti az operátori személyzet beavatkozását, így elkerülve a csövek helyi túlhevülését vagy a hőfeszültségekből adódó károkat [10]. Gázhűtésű reaktorokban fellépő anomáliák felismerésére,

illetve rezgésanalízisre alapozott csapágy degradáció meghatározására is sikerrel alkalmazták az RNN architektúrát [14]. A fenti két példa jól szemlélteti az RNN erősségét, amikor a magyarázó és magyarázott változó között időbeli eltérés van.

4.3. Konvolúciós neurális hálók alkalmazása

Atomerőművekben komoly nehézséget okoz olyan nem várt üzemzavarok előrejelzése, melyek akár a létesítmény paramétereinek tucatjaitól, illetve ezek változásaitól függenek, ráadásul az üzemzavar és a paraméterek közötti kapcsolat nem definiálható egyértelműen. Egy kutatás során olyan diagnosztikai célt szolgáló CNN készült, melynek pontossága a szerzők szerint lehetővé teszi annak valós idejű alkalmazását is. A probléma megoldása kifejezetten egyedi módon történt, mivel az atomerőmű 1004 vizsgált paraméterét, illetve ezen értékek múltbéli viselkedésmintáit képpé konvertálták az adatelőkészítés során, hogy a CNN hatékonyan tudja feldolgozni azokat. Minden adatot normalizáltak, így a 0 és 1 közötti értékek egy fehér-fekete színskálán voltak megjeleníthetőek [11].

5. A TERMODINAMIKAI MODELLRE ALAPOZOTT NEURÁLIS HÁLÓ ELKÉSZÍTÉSE

E fejezetben az általam készített termodinamikai modellt, az abból kinyert adatok kezelését, illetve a neurális háló optimalizálási folyamatát mutatom be.

5.1. A CycleTempo modell

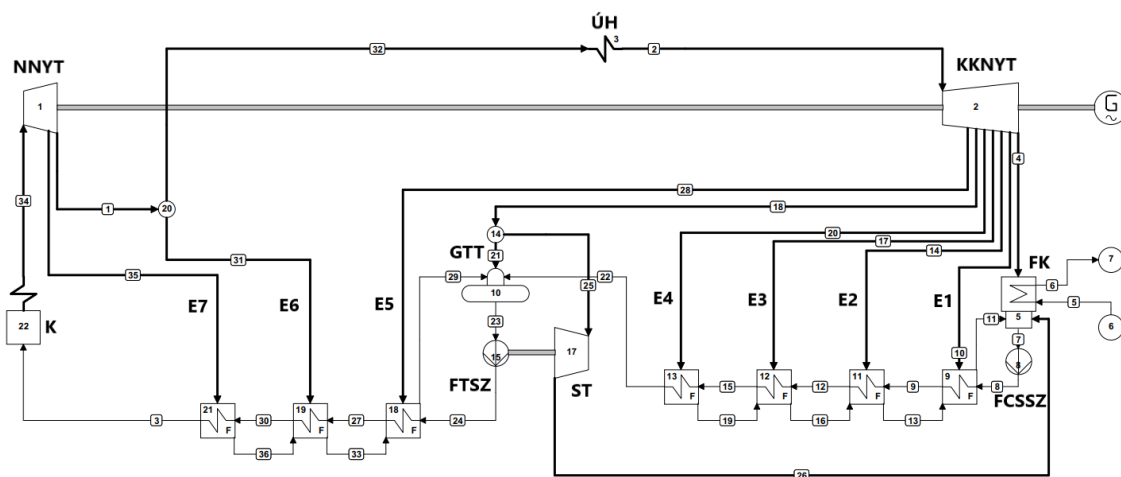
A szoftverben egy 800 MW névleges teljesítményű szuperkritikus szénérőmű [17] termodinamikai modelljét készítettem el tervezési és tervezettől eltérő üzemállapotokra. A turbinagépcsoport esetén mennyiségi szabályozást feltételeztem, így a nagynyomású szekciót 1 szabályzó fokozatú, 4 fűvókás berendezésként vettem fel (TU-CODE=21004). A közép- és kisnyomású részt egyházas kivitelként modelleztem. Az expanderek ikeráramúságát a párhuzamos áramok számával vettem figyelembe. Mivel a közép-kisnyomású rész 6 helyen is meg van csapolva, ezért az expanziós vonalhoz tartozó paramétert 1-nek választottam. A gépcsoport 3000 rpm fordulatszámon üzemel, így a turbinakód 52141-re adódott. A tápvíz-előmelegítő rendszer 7 fokozatú, kaszkád kapcsolásban. A fő tápvízszivattyú segéd turbinával hajtott, mely ugyanazon nyomásszinten kap gőzt, mint a gáztalanító táptartály. A kazánoldalt egy egyszerű „boiler” objektummal modelleztem. A tervezési állapothoz tartozó méretezési paramétereket a 5-1. táblázat, a hősemát a 5-1. ábra tartalmazza. A 5-2. ábra a tervezési üzemállapothoz tartozó Rankine-ciklust mutatja T-S diagramban ábrázolva.

5-1. táblázat: Tervezési üzemállapothoz tartozó paraméterek [17]

Létesítmény	Teljesítmény [MW]	800
Frissgőz paraméterek	Nyomás [bar]	290
	Hőmérséklet [°C]	600
Újrahevítés	Hőmérséklet [°C]	620
Kondenzátor	Üzemi nyomás [bar]	0,103
	Primer oldali nyomásesés [bar]	1
	Primer oldali ΔT [°C]	10
	Hűtővíznyomás [bar]	4
	Hűtővíz hőmérséklete [°C]	15
Kisnyomású TVE	E1 megcsapolási nyomás [bar]	0,43
	E2 megcsapolási nyomás [bar]	1,26
	E3 megcsapolási nyomás [bar]	3
	E4 megcsapolási nyomás [bar]	6,62
	DELTH [°C]	3
GTT	Nyomás [bar]	12

Nagynyomású TVE	E5 megcsapolási nyomás [bar]	28,9
	E6 megcsapolási nyomás [bar]	62,5
	E7 megcsapolási nyomás [bar]	92,2
	DELTL [°C]	5
	DELTH [°C]	0
Szivattyúk	Hatásfok [%]	85

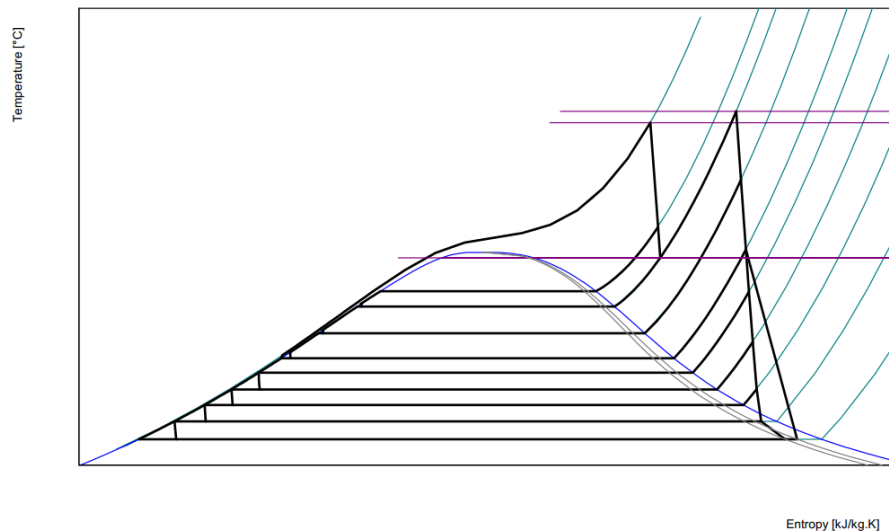
*A nem definiált nyomáseséseket elhanyagoltam.



NNYT: nagynyomású turbinaszekció
 ÚH: újrahevítő
 KKNYT: közép-kisnyomású turbinaszekció
 FK: főkondenzátor
 FCSSZ: főcsapadék-szivattyú
 K: kazán

E1...E4: kisnyomású TVE
 GTT: gáztalanító táptartály
 ST: segédturbina
 FTSZ: főtápvíz-szivattyú
 E5...E7: nagynyomású TVE

5-1. ábra: A 800 MW névleges teljesítményű szuperkritikus szénérőmű hősémája



5-2. ábra: A tervezési állapothoz tartozó Rankine-ciklus T-s diagramja

A geometriák rögzítése, illetve a Stodola-féle gőzkúpából számolható nyomásparaméterek (megcsapolások, újrahevítés, GTT) eltávolítása után a modellt tetszőleges részterhelésen tudtam futtatni a szoftver „Production functions” opciójának segítségével.

Egy erőmű üzemi paramétereit és hatásfoka az adott terhelési állapoton kívül frissvízhűtésű kondenzátor esetén a belépő hűtővíz hőmérsékletétől is függ, vagyis a tervezési üzemállapotot 800 MW-os teljesítmény, illetve 15 °C-os belépő hűtővíz hőmérséklet definiálja.

Különböző kondenzátornyomással paraméterezett modelleket hoztam létre annak érdekében, hogy a létesítmény környezeti hőmérséklettől függő viselkedését is le tudjam írni. Az eltérő hűtővíz hőmérsékletekhez tartozó kondenzátornyomást a (30), (31) összefüggések alapján határoztam meg.

$$T_{FK} = T_{sat} = T_{hvk} + \Delta T_H \quad (30)$$

$$p_{FK} = p_{sat}(T_{sat}) \quad (31)$$

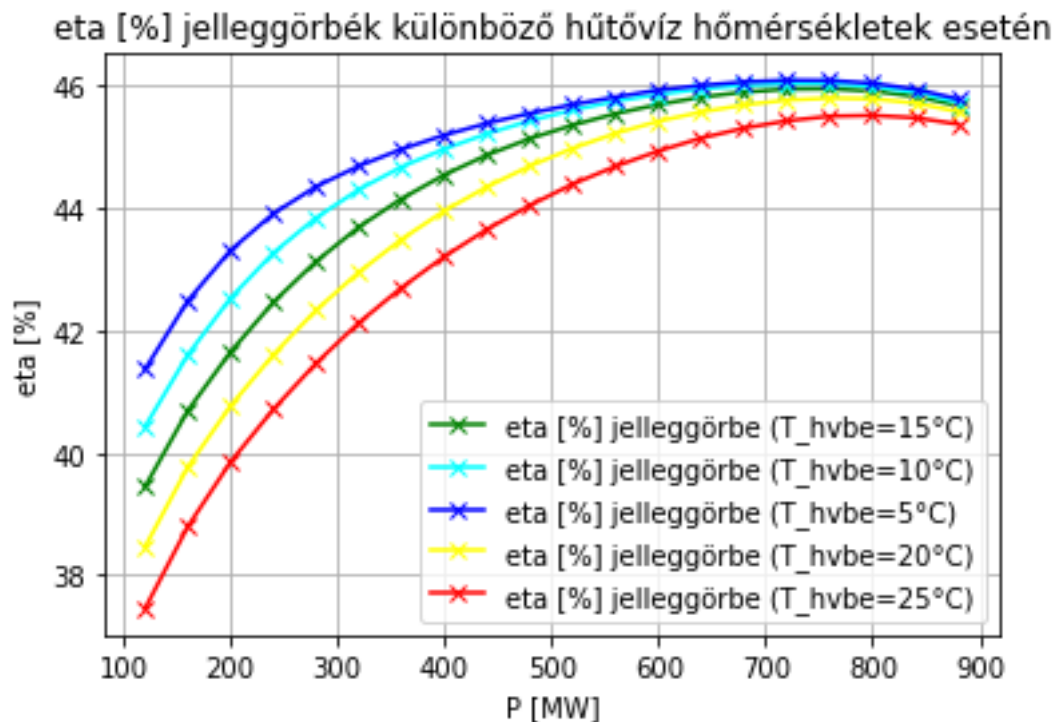
Ahol ΔT_H a 800 MW-os terhelési szinten és 15 °C-os belépő hűtővíz hőmérséklet mellett kialakuló hőfokrás, mely a hűtővíz hőmérséklet változása esetén állandó marad, hiszen ez a hőfokrás csak a kondenzátor felületétől függ. A tervezési hőmérsékleten kívül még négy esetet vizsgáltam.

5.2. A háló tanításához szükséges adathalmaz létrehozása

A háló bemeneti adatait a CycleTempo modell tervezési üzemállapoton kívüli sorozatos futtatásával nyertem. 15%-os és 110%-os terhelések között vettem fel az adatokat 5%-os lépcsőkkel, azaz 20 munkapontot rögzítettem az 5 °C, 10 °C, 15 °C, 20 °C, valamint 25 °C belépő hűtővíz hőmérséklettel paraméterezett modellek esetén, így összesen egy 100 pontból álló adatsor keletkezett. A munkapontok azon paramétereit változtattam ki, melyek a terhelés változtatásával szintén változnak. Ezek a következők voltak:

- Tápvíz hőmérséklete (T_{tv}) és tömegárama (m_{tv})
- Újrahevítési nyomás (p_{uh}) és tömegáram (m_{uh})
- Kondenzátorba belépő munkaközeg tömegárama (m_{kond})
- Megcsapolási nyomások (p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{E7})
- GTT nyomása (p_{gtt})

A hatásfok jelleggörbéket különböző belépő hűtővíz hőmérsékletekre a 5-3. ábra tartalmazza.

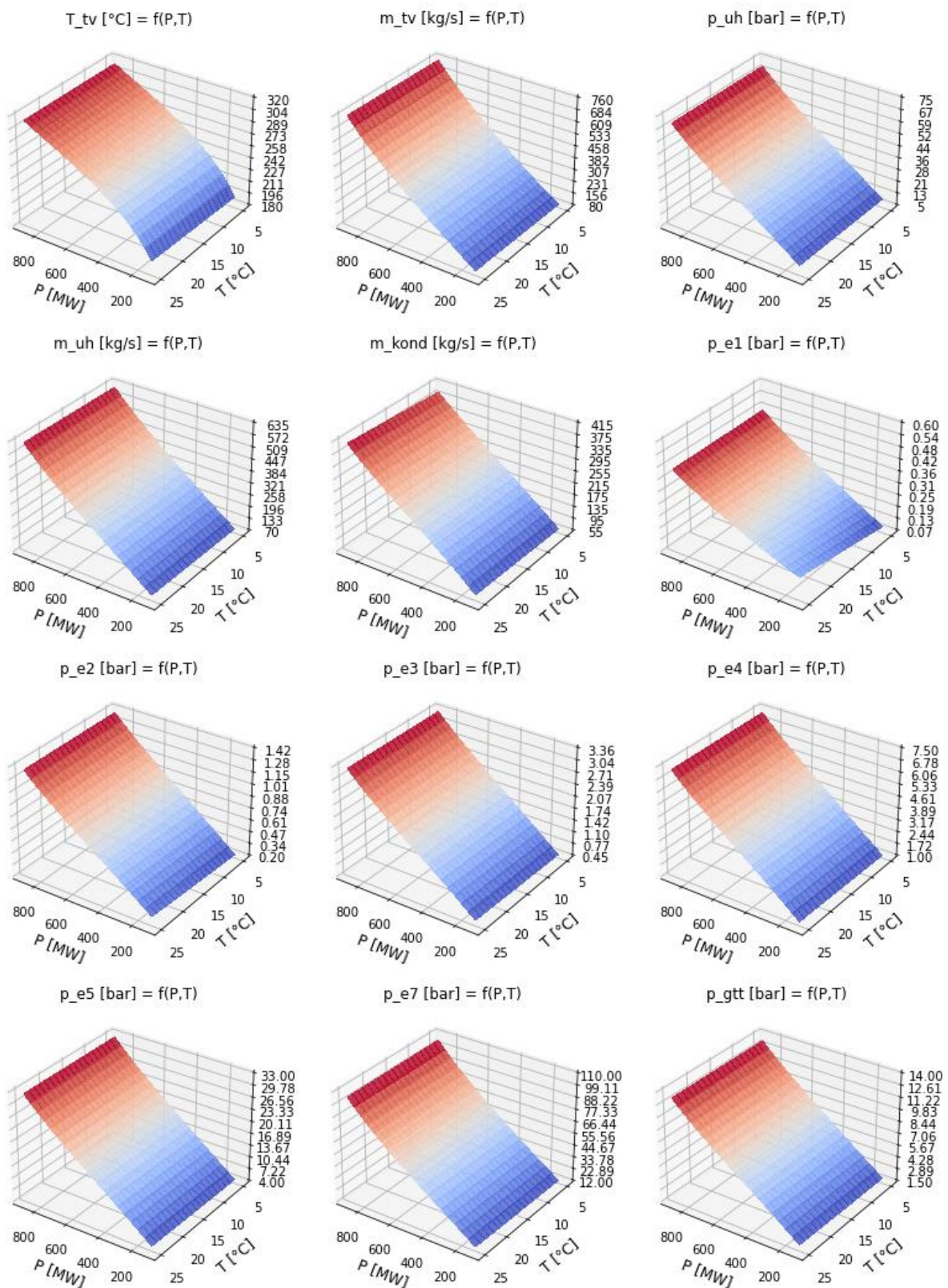


5-3. ábra: Hatásfok jelleggörbék a teljesítmény, illetve a hűtővíz belépő hőmérsékletek függvényében

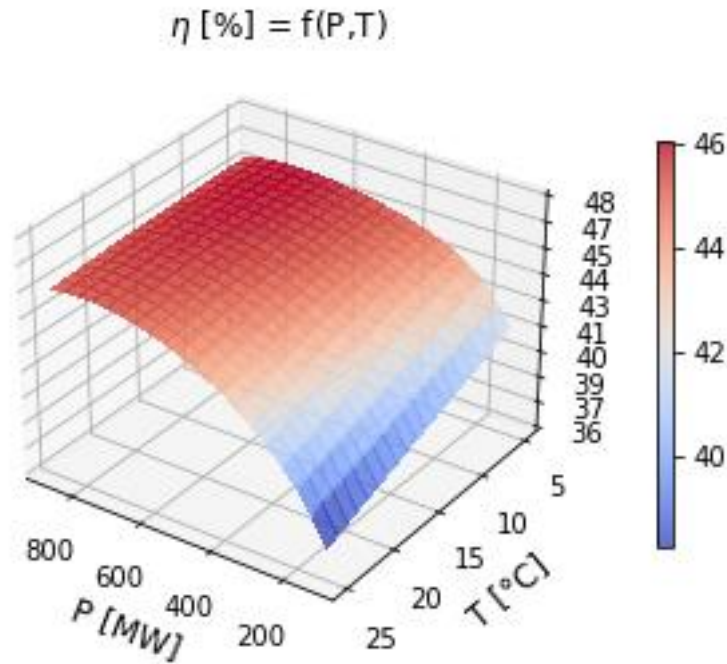
Ahhoz, hogy a háló minél pontosabb becslést tudjon adni a nettó hatásfokra, ennél nagyobb felbontású adathalmazra van szükség. A fenti paramétereket adott terhelési szint és különböző belépő hűtővíz hőmérsékletek mellett a Python „matplotlib” könyvtára segítségével ábrázoltam, majd a „numpy” könyvtár „poly1d” függvényének alkalmazásával első- és másodfokú görbékkel közelítettem. A kapott függvények együtthatóit külön változóknak tároltam, és az üzemi paramétereket a terhelési szinthez hasonlóan 20 pontból álló 5-25 °C-os belépő hűtővíz tartományban értékeltem ki, így az adathalmaz 100-ról 400 eleműre duzzadt. Itt fontos megjegyezni, hogy így az eredeti 5 munkapontbeli értékek is torzulnak, vagyis a háló az eredeti bemenő paraméterek alapján nem lesz képes a nettó hatásfok pontos becslésére. A képzett adatsor tehát 120-880 MW-os terhelési, illetve 5-25 °C-os belépő hűtővíz hőmérséklet tartományban tartalmazza az erőmű kiválasztott 12 db üzemi paraméterét, melyek 20x20-as mátrixokként is reprezentálhatóak.

A négyzetes mátrixok lehetővé teszik, hogy felületként jelenítsük meg az egyes üzemi paramétereket a már korábban említett „matplotlib” könyvtár segítségével, ahol minden egyes teljesítmény-környezeti hőmérséklet értékpárhoz, vagyis a P-T sík

adott pontjához az ábrázolt üzemi paraméter egy értéke tartozik. Az egyes üzemi paraméterek által definiált, P-T sík feletti felületeket a 5-4. ábra, míg a neurális háló által leírandó nettó hatásfok felületet a 5-5. ábra mutatja.



5-4. ábra: Üzemi paraméterek a teljesítmény és a belépő hűtővíz hőmérséklet függvényében



5-5. ábra: Nettó hatásfok a teljesítmény és a belépő hűtővíz hőmérséklet függvényében

5.3. Az adatok előkészítése

Először a 16 paramétert tartalmazó adatsorból elkülönítettem a nettó hatásfokot és azt egy vektorban tároltam el, majd a bemeneti jellemzők közül eltávolítottam a terhelést, mivel az a teljesítménnyel egyenesen arányos. Az előbbi műveletekhez a Python „Pandas” függvénykönyvtárát használtam. Ezután mind a 400x14-es bemeneti mátrixot, mind a hatásfokvektort rend szerint 75%-25% arányban tanító- és tesztalmazra bontottam a „scikit-learn” könyvtár segítségével. Az algoritmus az adatok sorrendjének randomizálását is elvégezte, ezzel a háló teljesítménye, robusztussága volt növelhető a tesztalmazon. Mivel a vizsgált paraméterek merőben eltérő értéktartományokban mozogtak, ezért szükséges volt az adatok standardizálása, hogy a mátrixszorzások után kialakult összegekben nagyjából egyenlő súllyal szerepeljenek a munkaponti jellemzők. Ezt a transzformációt szintén a „scikit-learn” könyvtárral valósítottam meg. A standardizált sokaság átlaga 0, szórása 1, és a (32) képlettel számolható [8], ahol x az adott paraméterhalmaz egy eleme, μ a paraméterhalmaz átlaga, σ pedig annak szórása.

$$z = \frac{x - \mu}{\sigma} \quad (32)$$

5.4. Különböző hálóarchitektúrák vizsgálata

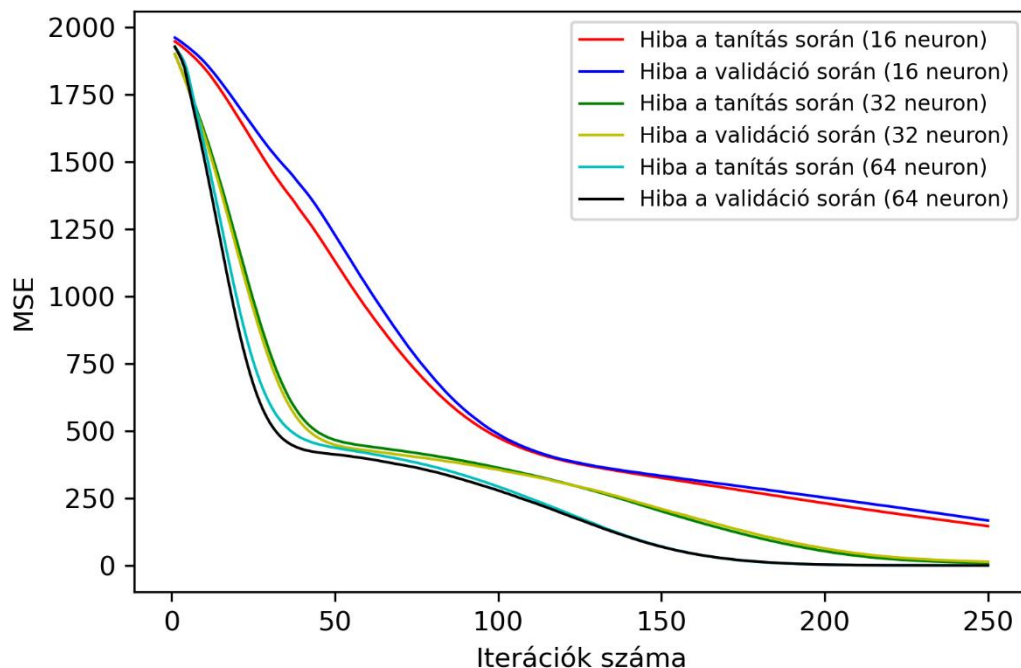
Vizsgálataim teljesen kapcsolt, egy, illetve két rejtett réteget tartalmazó hálókra terjedtek ki. Céлом volt megtalálni azt a hálóarchitektúrát, melynél a hibafüggvény konvergenciája relatíve gyorsan megvalósul, illetve amely a lehető legkevesebb modellparaméterrel a legpontosabb becslést adja a nettó hatásfokra a bemeneti paraméterek

alapján. Munkám során a rejtett rétegek, valamint rejtett rétegben lévő neuronok számával, illetve kétféle aktivációs függvényvel kísérleteztem. A modelleket a „Keras” gépi tanulási függvénykönyvtár segítségével hoztam létre.

5.4.1. EGY REJTETT RÉTEGET TARTALMAZÓ ARCHITEKTÚRÁK

5.4.1.1. Architektúrák ReLU aktivációs függvényekkel

Első közelítésben a rejtett és kimeneti rétegekben egyaránt *ReLU* aktivációs függvényt használtam. Három különböző hálót tanítottam be, melyek rejtett rétegei rend szerint 16, 32, 64 neuront tartalmaztak, az algoritmusok **250 iteráción keresztül futottak**. A teszt adatsor 20%-át validációs halmazként különítettem el, így a háló párhuzamosan számolta az adott iteráción belüli modellparaméterek mellett a kimeneteket, valamint tárolta a négyzetes hibaösszeget, és az átlagos abszolút eltérés mutatókat minden lépésre. A négyzetes hibaösszegek időbeli alakulását a tanulás során a „matplotlib.pyplot” könyvtár segítségével ábrázoltam. A tanítási és validációs hibát az iterációs szám függvényében jellegre helyesen a 5-6. ábra mutatja be a három *ReLU* aktivációs függvényt használó modell esetére.



5-6. ábra: Tanítási és validációs hiba az iterációk függvényében (egy rejtett réteg, ReLU)

A 5-6. ábra is jól mutatja, hogy a négyzetes hibaösszeg a tanítási és validációs adattal számítva is ugyanolyan jelleggel csökken, a görbék széttartása, így a háló túlulása nem tapasztalható.

A hálók tanítását és kiértékelését 100 alkalommal végeztem el, mivel az eredmények minden egyes futtatás esetén eltérőek attól függően, hogy a súlymátrixok, illetve eltolás vektorok milyen random értéken vannak inicializálva. A háló eredményességét Excel-ben elemeztem. A prognózis pontosságának mutatóiként a determinációs együtthatót² (R^2), valamint az átlagos négyzetes hibaösszeget (MSE) választottam. A determinációs együttható 100 futtatásra vett átlagát, szórását és maximum értékét adtam meg a táblázatban. A determinációs együtthatót a (33) képlettel számítottam, ahol y a magyarázott változó, vagyis a nettó hatásfok várt értéke, $\hat{y}$ pedig a modell által szolgáltatott prognózis.

$$R^2 = \frac{\sum(\hat{y} - \bar{\hat{y}})^2}{\sum(\hat{y} - \bar{\hat{y}})^2 + \sum(y - \bar{y})^2} \quad (33)$$

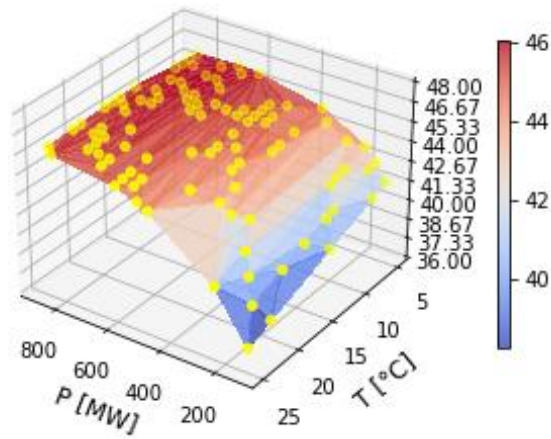
A 5-2. táblázat az eredményességi mutatókon kívül tartalmazza az adott hálóarchitektúrához tartozó modellparaméterek (p), valamint a konvergenciához szükséges iterációk számát (i). A konvergenciához szükséges iterációk száma azt mutatja meg, hogy mely tanulási ciklusban érte el a hibafüggvény (MSE) a 10 alatti értéket. A „div” oszlopban az olvasható le, hogy a 100 futtatásból hány alkalommal volt divergens a háló. A hálót akkor értékeltem divergensként, ha a 250 tanítási ciklus után a hibafüggvény 500 feletti volt. A 5-7. ábra az egyes modellek 100 futtatás közötti legjobb becsléseit jeleníti meg a teszt adatsor $f(P;T)$ diagramja alatt. A 5-8. ábra a becslések relatív hibáit ábrázolja szintén $f(P;T)$ diagramon, továbbá a becslés eltéréseinek gyakoriságát hisztogramon szemléltettem. Viszonyításképp a hisztogramon a normális eloszláshoz tartozó görbét is kirajzoltam. A táblázat, illetve az ábra alapján is látszik, hogy a felsorolt három architektúra közül a 64 rejtett rétegbeli neuront tartalmazó a legpontosabb, azonban a háló által megvalósított regresszió így is gyenge becsléseket biztosít, mert a hatásfok által definiált felületet még csak jellegre helyesen sem tudja leírni a háló. Az architektúra további tökéletesítése volt szükséges.

5-2. táblázat: Egy rejtett rétegű, ReLU aktivációs függvényeket használó modellek jellemzői

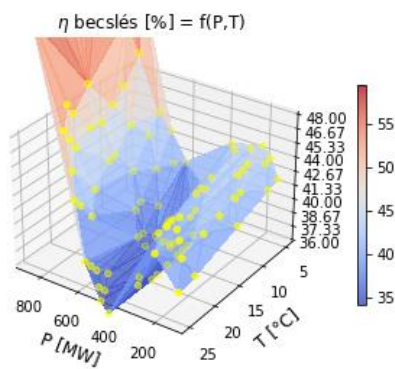
	f_{r1}	n_{r1}	f_{r2}	n_{r2}	f_k	p	R^2			MSE _{átlag}	I _{átlag}	div
							$R^2_{\text{átlag}}$	σ_{R^2}	MAX(R^2)			
1	ReLU	16	-	-	ReLU	257	0,479	0,011	0,502	135,588	250,000	7
2	ReLU	32	-	-	ReLU	513	0,512	0,068	0,698	15,748	244,066	9
3	ReLU	64	-	-	ReLU	1025	0,689	0,078	0,839	2,555	182,927	4

² A determinációs együttható arra ad választ, hogy a magyarázó változó a magyarázott változó szórádását (variánciáját) milyen hányadban határozza meg [18].

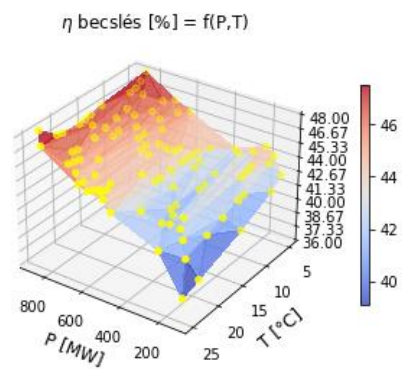
$$\eta \text{ teszt [\%]} = f(P, T)$$



16 neuron

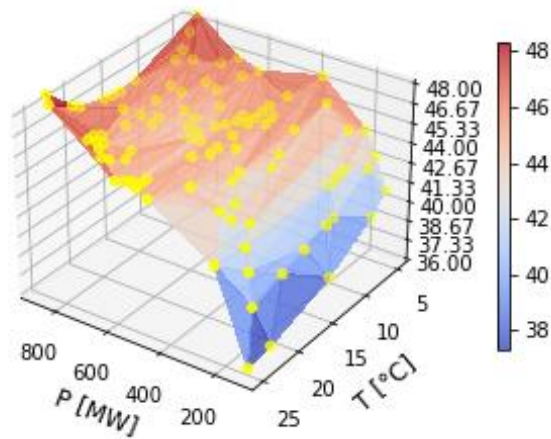


32 neuron

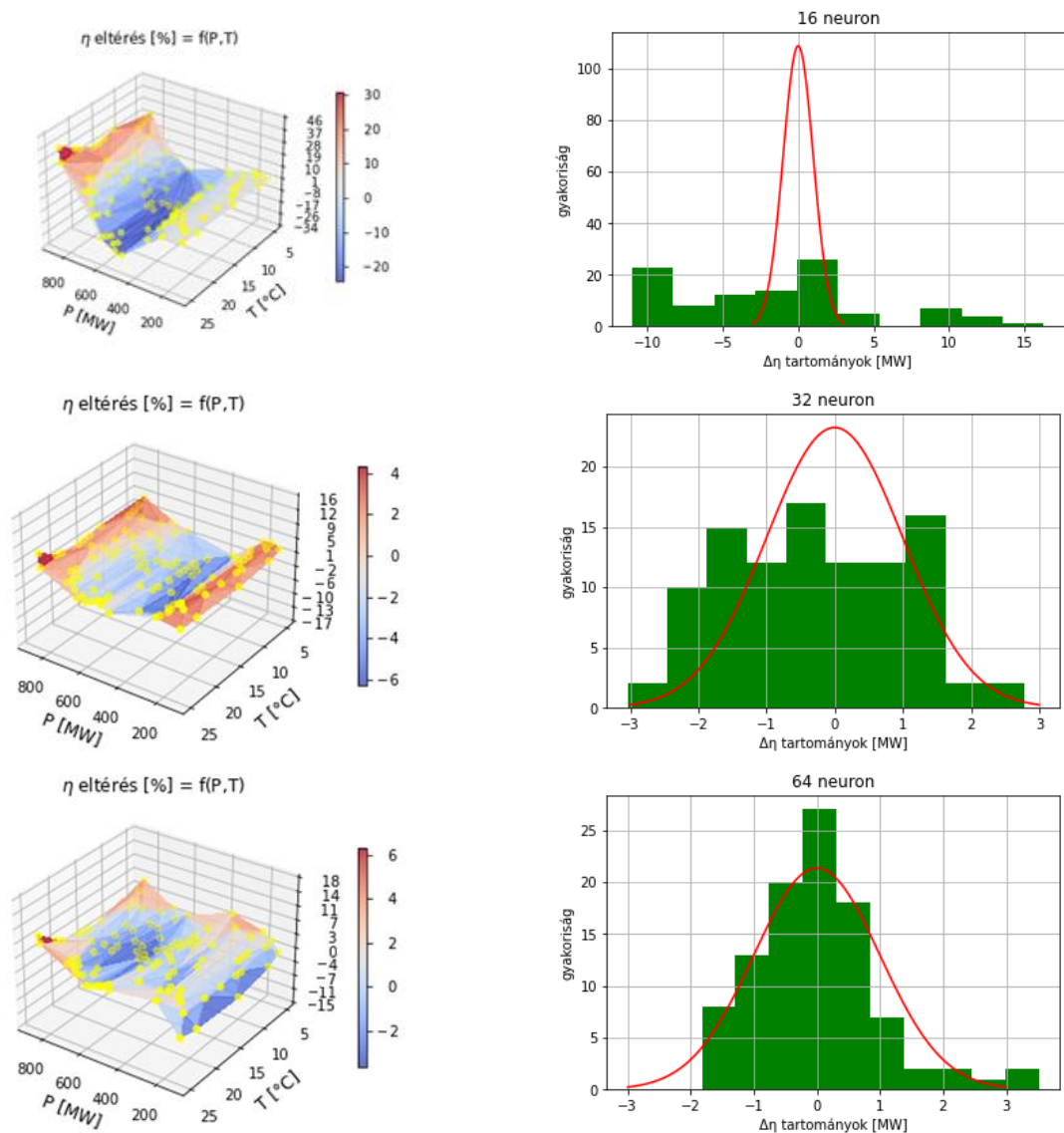


64 neuron

$$\eta \text{ becslés [\%]} = f(P, T)$$



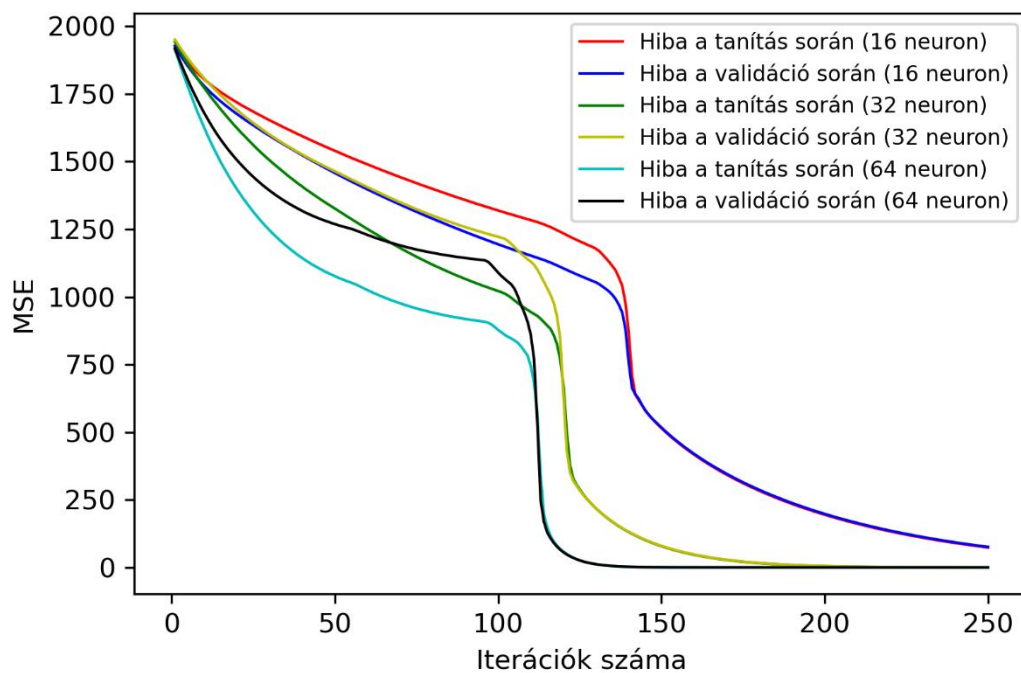
5-7. ábra: Egy rejtett rétegű, *ReLU* aktivációs függvényvel rendelkező architektúrák becslései



5-8. ábra: Modellbecslések relatív hibái (bal oldal) és eltérések hisztogramjai (jobb oldal) – egy rejtett réteg, *ReLU*

5.4.1.2. Architektúrák *tanh* aktivációs függvényekkel

A vizsgálatok következő lépéseként a rejtett réteg aktivációs függvényét a szigmoid görbe jellegű *tanh*-ra módosítottam, a kimeneti réteg aktivációs függvénye továbbra is *ReLU* maradt. Három különböző hálót tanítottam be, melyek rejtett rétegei rend szerint 16, 32, 64 neuront tartalmaztak. A hálók tanítására és értékelésére a 5.4.1.1 fejezetben leírtak érvényesek. A tanítási és validációs hibát az iterációs szám függvényében jellegre helyesen a 5-9. ábra mutatja be a három *tanh* aktivációs függvényt használó modell esetére.



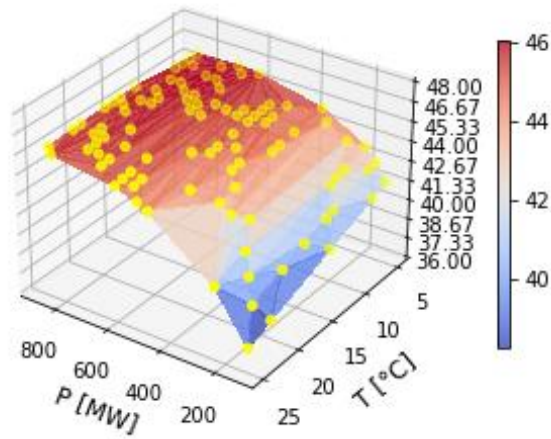
5-9. ábra: Tanítási és validációs hiba az iterációk függvényében (egy rejtett réteg, $\tanh$)

A 5-3. táblázat a $\tanh$ aktivációs függvényt alkalmazó modellek jellemzőit tartalmazza. Az 5-10. ábra az egyes modellek 100 futtatás közül legjobb becsléseit jeleníti meg a teszt adatsor $f(P;T)$ diagramja alatt. A 5-11. ábra a becslések relatív hibáit ábrázolja szintén $f(P;T)$ diagramon, továbbá a becslés eltéréseinek gyakoriságát hisztogrammon szemléltettem.

5-3. táblázat: Egy rejtett rétegű, $\tanh$ aktivációs függvényt használó modellek jellemzői

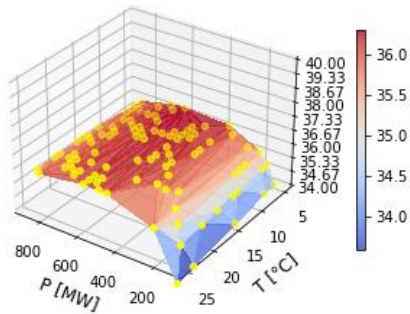
	f_{r1}	n_{r1}	f_{r2}	n_{r2}	f_k	p	R^2			$MSE_{\text{át-lag}}$	$\bar{i}_{\text{átlag}}$	div
							$R^2_{\text{átlag}}$	σ_{R^2}	$MAX(R^2)$			
4	$\tanh$	16	-	-	ReLU	257	0,494	0,000	0,495	85,380	250,000	0
5	$\tanh$	32	-	-	ReLU	513	0,924	0,013	0,953	0,331	187,300	0
6	$\tanh$	64	-	-	ReLU	1025	0,938	0,009	0,963	0,322	131,350	0

$$\eta_{\text{teszt}} [\%] = f(P, T)$$



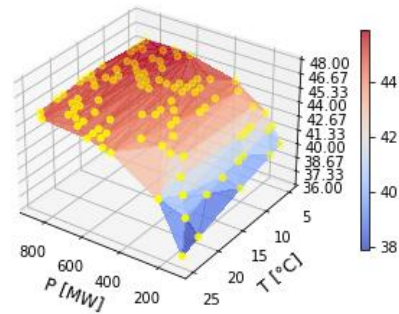
16 neuron

$$\eta_{\text{becslés}} [\%] = f(P, T)$$



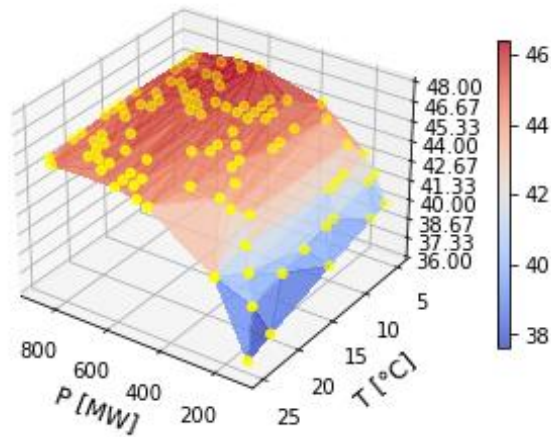
32 neuron

$$\eta_{\text{becslés}} [\%] = f(P, T)$$

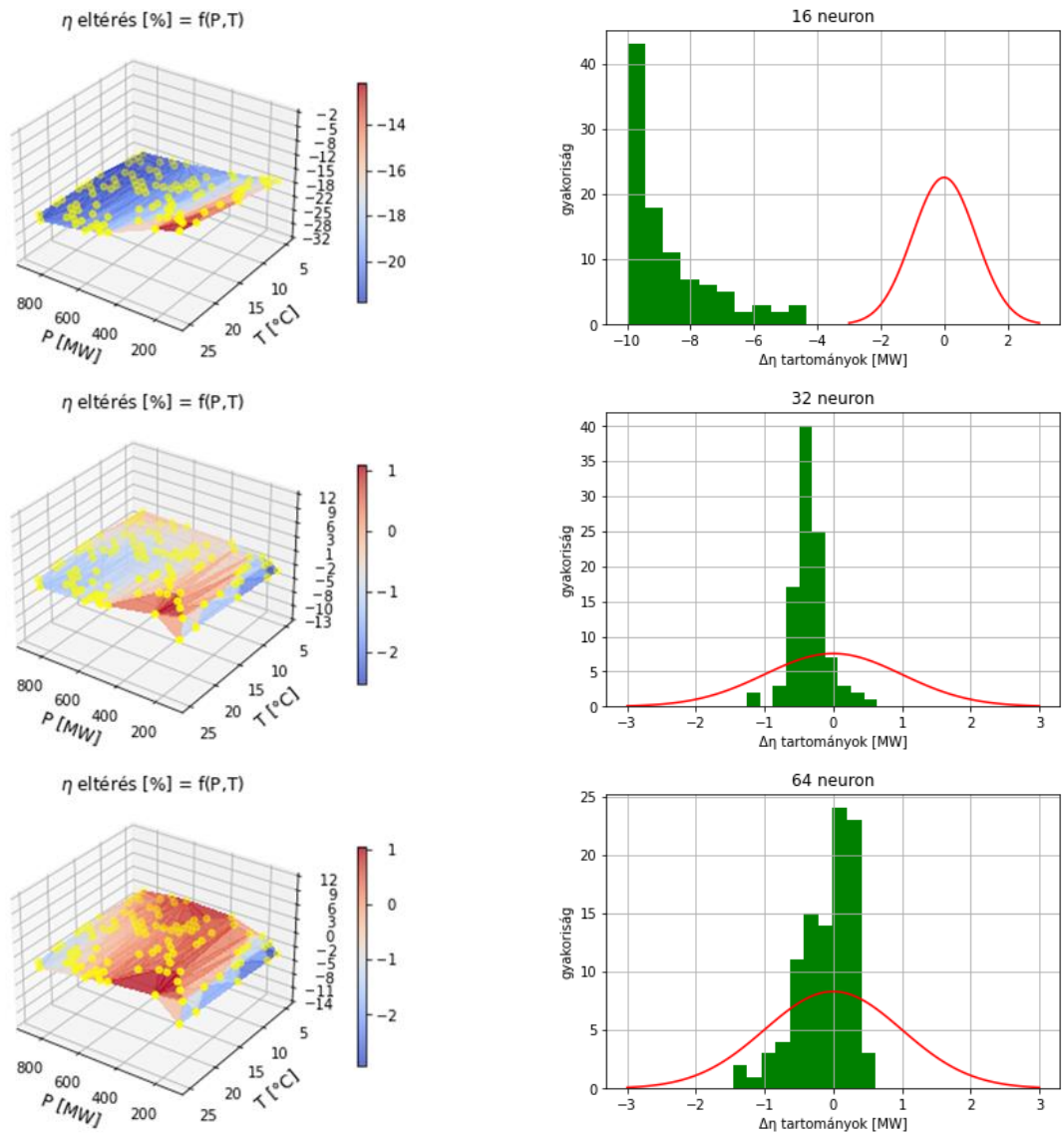


64 neuron

$$\eta_{\text{becslés}} [\%] = f(P, T)$$



5-10. ábra: Egy rejtett rétegű, tanh aktivációs függvényvel rendelkező architektúrák becslései



5-11. ábra: Modellbecslések relatív hibái (bal oldal) és eltérések hisztogramjai (jobb oldal) – egy rejtett réteg, $\tanh$

A rejtett rétegben $\tanh$ aktivációs függvényt alkalmazó, 32 és 64 neuronnal rendelkező modellek a táblázat, illetve az ábrák alapján már elfogadható becsléseket szolgáltatnak, valamint a háló konvergenciájához szükséges iterációk száma is csökkent. A 100 tanítás és kiértékelés során kapott eredmények sokkal kevésbé szórnak, vagyis ez az architektúra kevésbé érzékeny a súlymátrixok és eltolásvektorok inicializálására. A modellek minden esetben konvergensek voltak. Bár a 64 neuront tartalmazó háló a legpontosabb, az kétszer annyi modellparamétert tartalmaz, valamint a relatív hiba sem olyan kiegyenlített a P-T síkon, mint a 32 neuronnal ellátott modell esetén.

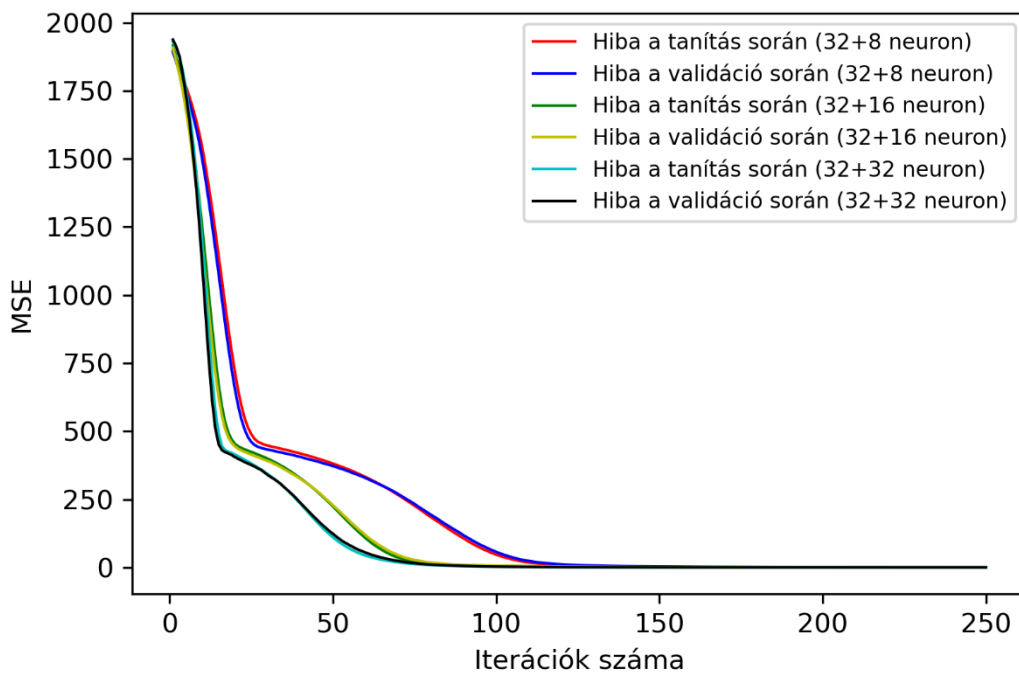
5.4.2. KÉT REJTETT RÉTEGET TARTALMAZÓ ARCHITEKTÚRÁK

A modellek viselkedését a rejtett rétegek számának növelése következtében e fejezetben vizsgáltam.

5.4.2.1. Architektúrák ReLU aktivációs függvényekkel

Elsőként a rejtett és kimeneti rétegekben egyaránt *ReLU* aktivációs függvényeket használtam. Az első rejtett rétegben minden esetben 32 neuronnal, míg a második rétegben 8, 16, illetve 32 neuronnal kísérleteztem. Ilyen módon a modellparaméterek száma összehasonlítható maradt az előző fejezetekben tárgyalt modellekével.

A hálók tanítására és értékelésére a 5.4.1.1 fejezetben leírtak érvényesek. A tanítási és validációs hibát az iterációs szám függvényében jellegre helyesen a 5-12. ábra mutatja be a három *ReLU* aktivációs függvényt használó modell esetére.

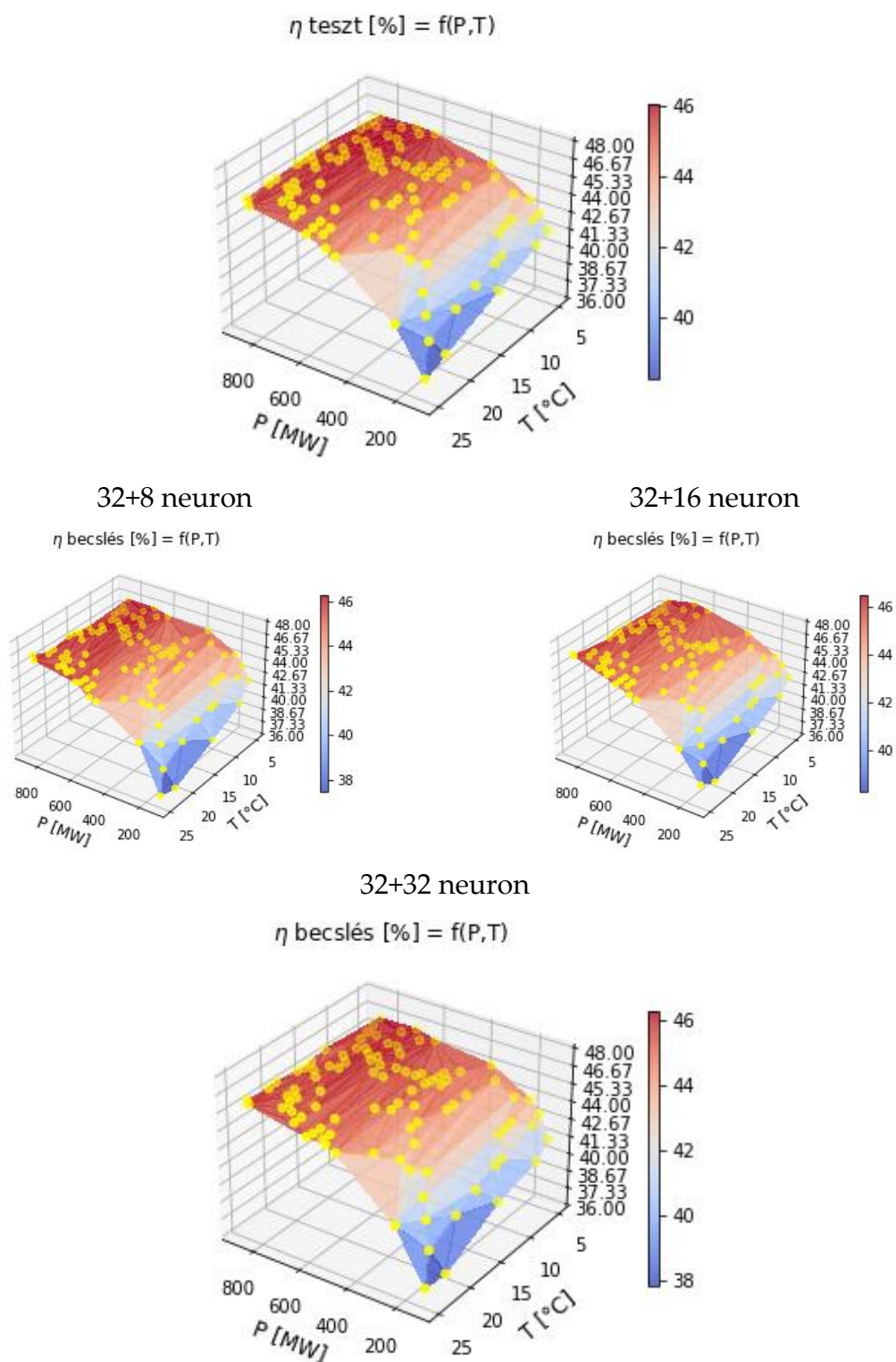


5-12. ábra: Tanítási és validációs hiba az iterációk függvényében (két rejtett réteg, ReLU)

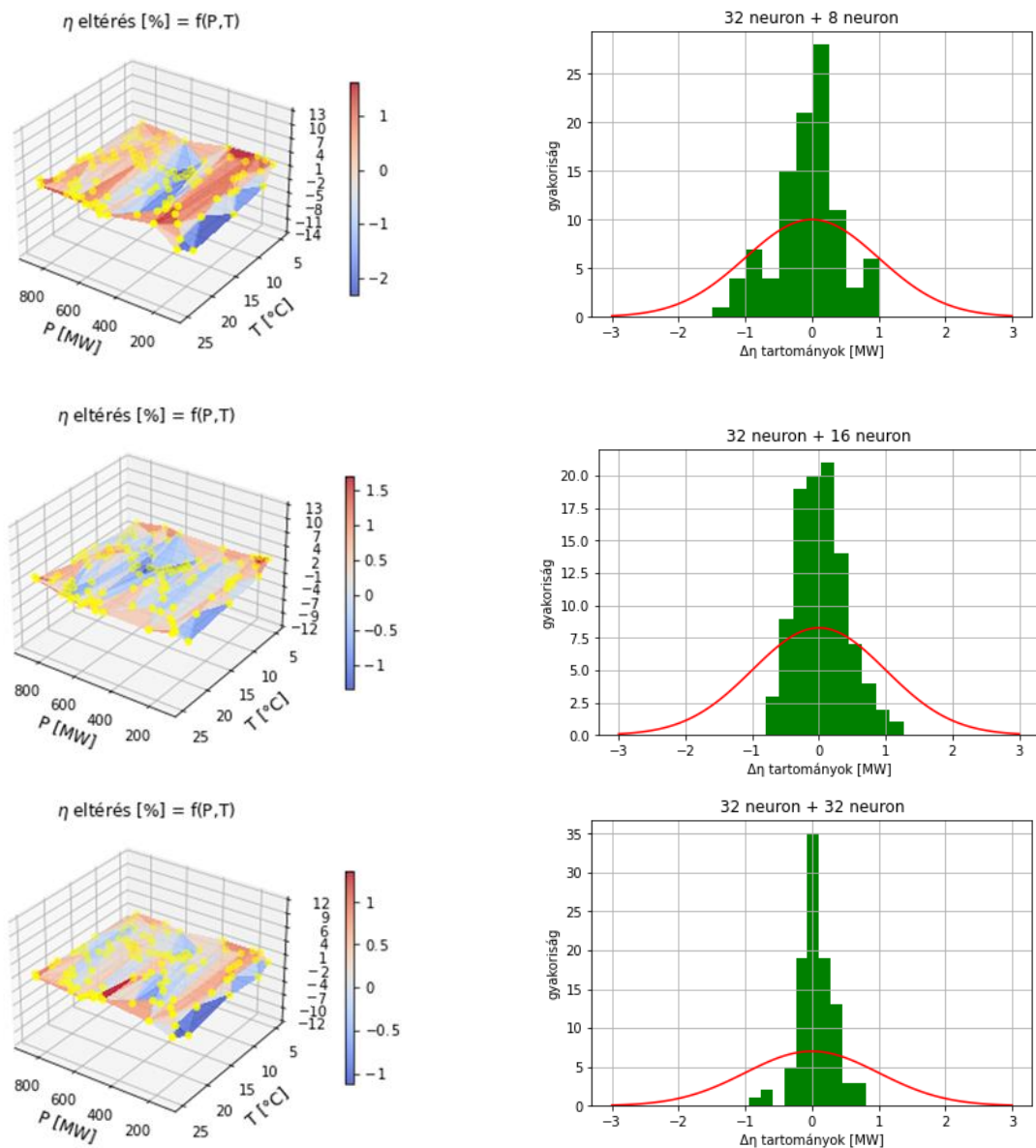
A 5-4. táblázat 5-3. táblázat a *ReLU* aktivációs függvényt alkalmazó két rejtett rétegű modellek jellemzőit tartalmazza. A 5-13. ábra az egyes modellek 100 futtatás közötti legjobb becsléseit jeleníti meg a teszt adatsor $f(P;T)$ diagramja alatt. A 5-14. ábra a becslések relatív hibáit ábrázolja szintén $f(P;T)$ diagramon, továbbá a becslés eltéréseinek gyakoriságát hisztogramon szemléltettem.

5-4. táblázat: Két rejtett rétegű, *ReLU* aktivációs függvényt használó modellek jellemzői

	f_{r1}	n_{r1}	f_{r2}	n_{r2}	f_k	p	R^2			$MSE_{\text{átlag}}$	$\bar{I}_{\text{átlag}}$	div
							$R^2_{\text{átlag}}$	σ_{R^2}	$MAX(R^2)$			
7	ReLU	32	ReLU	8	ReLU	721	0,857	0,053	0,945	0,733	116,239	12
8	ReLU	32	ReLU	16	ReLU	993	0,915	0,029	0,960	0,371	93,378	18
9	ReLU	32	ReLU	32	ReLU	1537	0,950	0,015	0,981	0,211	78,966	13



5-13. ábra: Két rejtett rétegű, ReLU aktivációs függvénnel rendelkező architektúrák becslései



5-14. ábra: Modellbecslések relatív hibái (bal oldal) és eltérések hisztogramjai (jobb oldal) – két rejtett réteg, ReLU

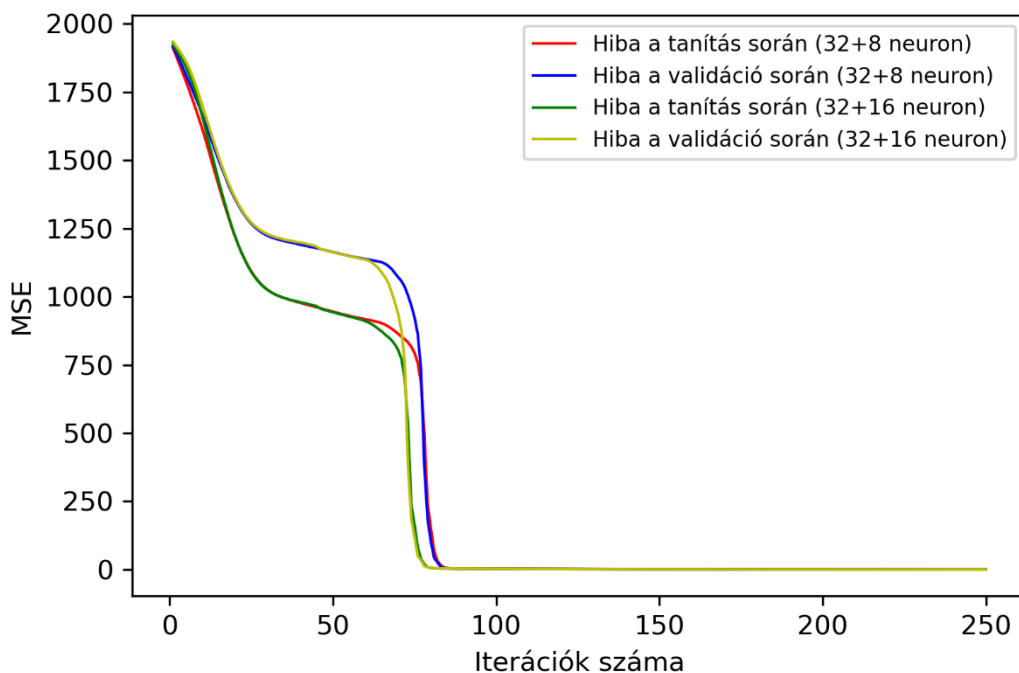
A mindkét rejtett rétegében 32 neuront és *ReLU* aktivációs függvényt tartalmazó modell az eddigiekben tárgyalt modellek közül a legpontosabb, azonban újabb 500 modellparaméter beemelését igényelte a determinációs együttható észrevehető növelése, ráadásul a 100 futtatásból 13 esetben a modell nem konvergált, hasonlóan a második rejtett rétegben 8, illetve 16 neuront tartalmazó esethez. Ki kell emelni, hogy a becslés pontossága jobban szórt a 100 futtatás esetén az egy rejtett rétegű, *tanh* aktivációs függvényt alkalmazó hálózhoz képest.

Elmondható tehát, hogy a modell robusztussága csökkent, hiszen az a rejtett rétegek növelésével érzékenyebbé vált a súlymátrixok és eltolásvektorok kezdeti értékeire. Azon esetekben viszont, amikor a modell konvergensenek bizonyult, akkor a tanulás sokkal gyorsabb ütemben zajlott le. Az egy rejtett réteget tartalmazó modellekhez viszonyítva nagyjából fele annyi iteráció volt szükséges a hibafüggvény 10 alatti értékének eléréséhez.

5.4.2.2. Architektúrák $\tanh$ aktivációs függvényekkel

Hasonlóan az egy rejtett réteget tartalmazó architektúrához, ez esetben is az első rejtett réteg aktivációs függvényének $\tanh$ -ra való módosításával kezdtem vizsgálataimat. Az első rejtett rétegben minden esetben 32 neuronnal, míg a második rétegben csupán 8 és 16 neuronnal kísérleteztem, mivel a háló pontossága nem javult jelentősebb mértékben a 16 neuront tartalmazó esetben, ahogy az a későbbi eredmények alapján látható lesz.

A hálók tanítására és értékelésére a 5.4.1.1 fejezetben leírtak érvényesek. A tanítási és validációs hibát az iterációs szám függvényében jellegre helyesen a 5-15. ábra mutatja be a három $ReLU$ aktivációs függvényt használó modell esetére.



5-15. ábra: Tanítási és validációs hiba az iterációk függvényében (két rejtett réteg, $\tanh$)

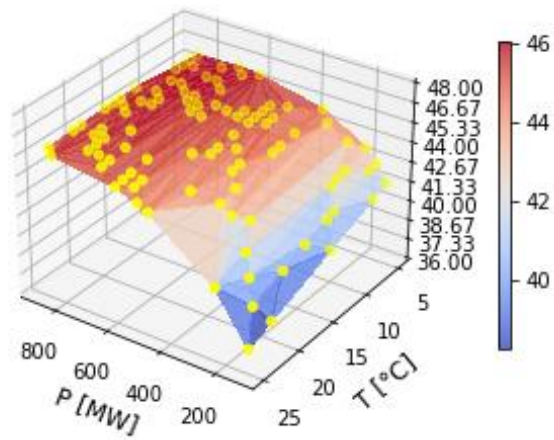
A 5-5. táblázat a $\tanh$ aktivációs függvényt alkalmazó két rejtett rétegű modellek jellemzőit tartalmazza. A 5-16. ábra az egyes modellek 100 futtatás közüli legjobb becsléseit jeleníti meg a teszt adatsor $f(P;T)$ diagramja alatt. A 5-17. ábra a becslések relatív hibáit ábrázolja szintén $f(P;T)$ diagramon, továbbá a becslés eltéréseinek gyakoriságát hisztogramon szemléltettem.

5-5. táblázat: Két rejtett rétegű, $ReLU$ aktivációs függvényt használó modellek jellemzői

	f_{r1}	n_{r1}	f_{r2}	n_{r2}	f_k	p	R^2			$MSE_{\text{átlag}}$	$\bar{I}_{\text{átlag}}$	div
							$R^2_{\text{átlag}}$	σ_{R^2}	$MAX(R^2)$			
10	$\tanh$	32	$ReLU$	8	$ReLU$	721	0,617	0,203	0,976	18,319	203,956	10
11	$\tanh$	32	$ReLU$	16	$ReLU$	993	0,597	0,168	0,980	10,628	205,344	7

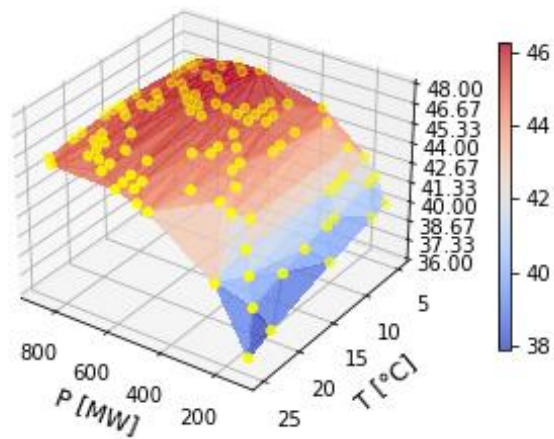
32+8 neuron

$$\eta_{\text{teszt}} [\%] = f(P, T)$$



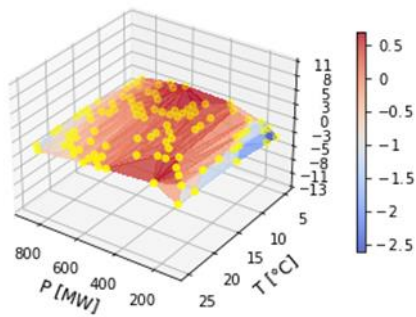
32+16 neuron

$$\eta_{\text{becslés}} [\%] = f(P, T)$$

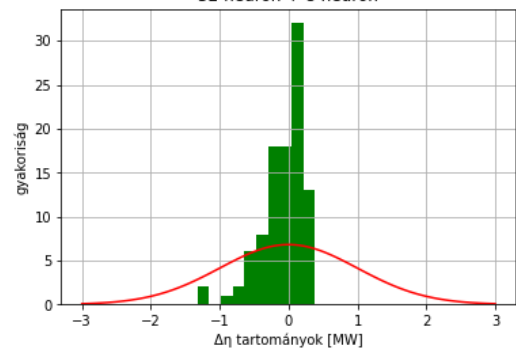


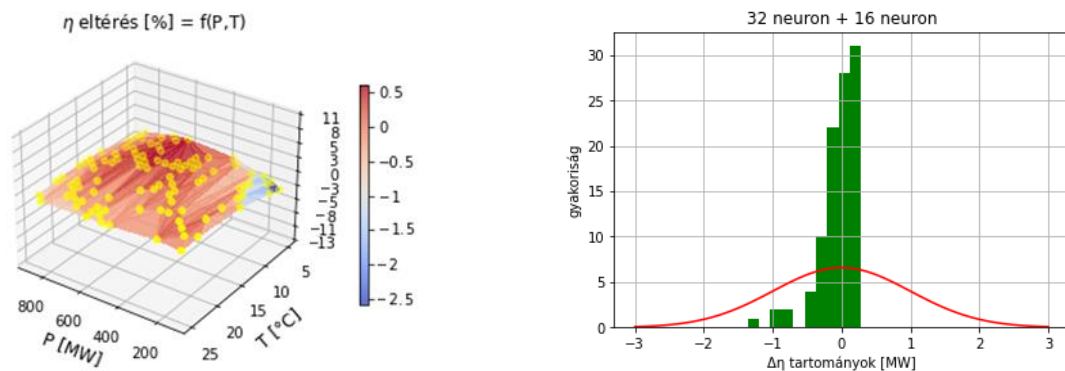
5-16. ábra: Két rejtett rétegű, $\tanh$ aktivációs függvényvel rendelkező architektúrák becslései

$$\eta_{\text{eltérés}} [\%] = f(P, T)$$



32 neuron + 8 neuron





5-17. ábra: Modellbecslések relatív hibái (bal oldal) és eltérések hisztogramjai (jobb oldal) – két rejtett réteg, $\tanh$

A kizárólag $ReLU$ aktivációs függvényeket alkalmazó két rejtett rétegű modellekkel összehasonlítva a jelen fejezetben tárgyalt modellek csupán konvergenciájuk és a legjobb becsléseik szempontjából teljesítenek jobban. Az eredmények szórása, átlagos determinációs együtthatója, valamint a tanulás gyorsasága egyaránt nagy mértékben csökkent. Az eddigiekben tárgyalt modellek közül ezen hálóarchitektúra a legérzékenyebb a modellparaméterek inicializált értékeire, így ezen modellek minősíthetőek a legkevésbé robusztusnak.

Ha a háló első és második rejtett rétegében is $\tanh$ -ra cseréljük az aktivációs függvényt, valamint első rejtett rétegében 32 neuronnal, a másodikban 8 neuronnal láttam el az architektúrát, akkor a modell nagyon rosszul teljesített a teszt adatsoron. Mind a 100 esetben 0,5 alatti determinációs együtthatót produkál, illetve 68 esetben nem is konvergens, ezért erre a hálóarchitektúrára vonatkozó eredményeket részletesen nem közlöm.

5.4.3. TAPASZTALATOK ÖSSZEFOGLALÁSA

Az előző alfejezetekben tárgyalt modellek közül három architektúra szolgáltatott elfogadható pontosságú becslést, relatíve kis szórás és kevés számú divergens eset mellett.

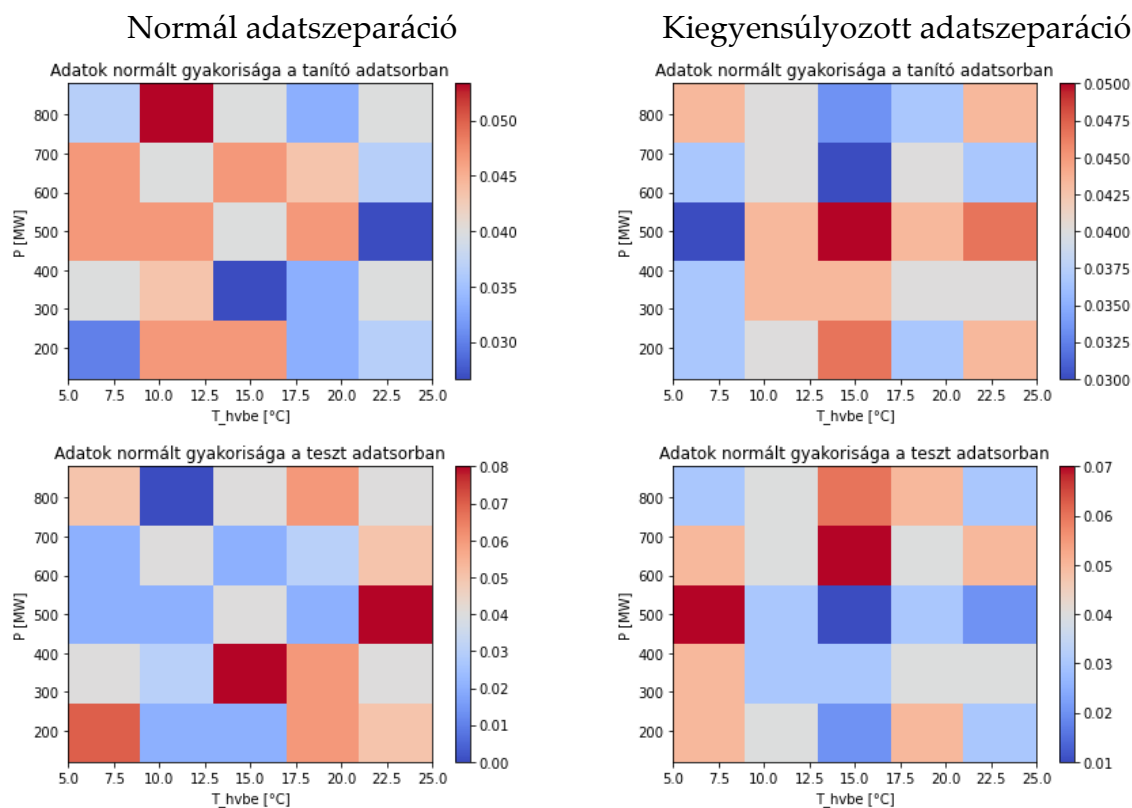
Az átlagosan legpontosabb és a tanulási ciklusok során a leggyorsabb modellnek a két rejtett rétegű, 32-32 neuront, illetve $ReLU$ aktivációs függvényeket tartalmazó háló bizonyult, amely viszont egyes esetekben nem volt konvergens.

Az előzőtől átlagos pontosság és tanulási gyorsaság szempontjából egy kissé elmaradnak az egy rejtett rétegű, $\tanh$ aktivációs függvényt alkalmazó, 32 és 64 neuront tartalmazó hálók, azonban ezt az eredményt 500-zal és 1000-rel kevesebb modellparaméterrel érték el, továbbá mind a 100 esetben konvergensnek bizonyultak. Ezen két hálóarchitektúra közül a 64 neuront tartalmazó modell minősíthető a legrobusztusabbnak, mert a determinációs együttható szórása ez esetben volt a legkisebb.

5.5. A tanító és teszt adatsor kiegyensúlyozása

Megvizsgáltam, hogy az algoritmus, amikor random módon a kiinduló adatsort felosztja tanító- és teszt-halmazokra, akkor azt milyen teljesítmény- és hőmérséklettartomány szerinti megoszlásban teszi. Feltételeztem, hogyha a P-T sík egyes cellái arányában ugyanannyi adatpontot tartalmaznak a tanító és teszt adatsorban, akkor a modell becslőképessége javulni fog. Vagyis a háló ebben a megközelítésben akkor szolgáltatná a legpontosabb becslést, ha az egyes tartományokhoz tartozó normált gyakoriságok a tanító és teszt-halmazokra megegyeznének. A normált gyakoriságokat szemléltető diagramokat a „matplotlib” könyvtár „hist2d” függvénye segítségével hoztam létre.

Az adatok eloszlását úgy próbáltam kiegyenlíteni, hogy a 400 pontból álló, randomizált halmazt az eredetitől eltérően nem egészében osztottam 75-25% arányban tanító és teszt adatsorokra, hanem 20-as egységenként végeztem el a szétválasztást. Ezt az eljárást értettem a „tanító és teszt adatsor kiegyensúlyozása” alatt. A tanító és teszt-halmazokban lévő adatok normált gyakoriságait szemléltető diagramok összehasonlítását a két adatszeperációs eljárás esetén a 5-18. ábra tartalmazza.



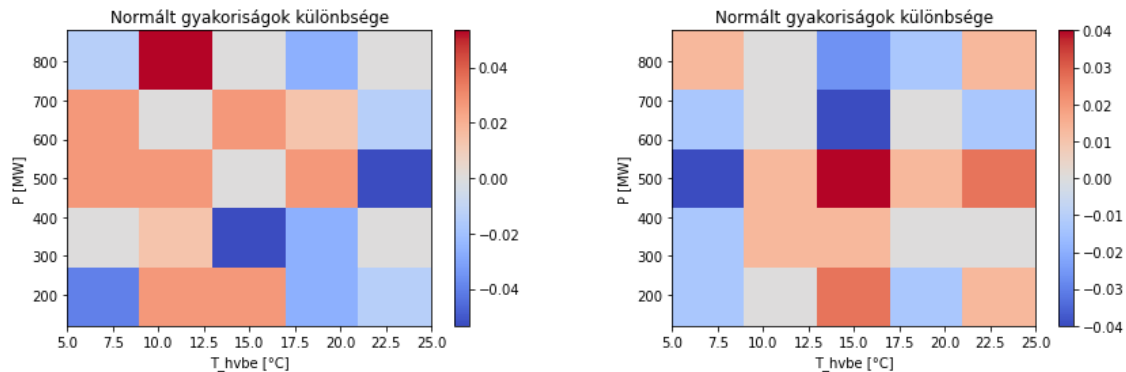
5-18. ábra: Tanító és teszt-halmazokban lévő adatok normált gyakoriságai a két adatszeperációs eljárás esetén

A két-két mátrixból egy különbség mátrixot képeztem olyan módon, hogy tanító adatsor normált gyakoriságaiból elemenként kivontam a teszt adatsor normált gyakoriságait. Ha a különbség mátrix adott elemének értéke negatív, akkor a megfelelő P-T tartományon az adatok túl vannak reprezentálva a teszt-halmazban, vagyis a modell alulillesztett lesz várhatóan. Ha pedig a különbség mátrix adott eleme pozitív, akkor

a vizsgált P-T cellában a tanító halmaz adatai vannak túlsúlyban, vagyis várhatóan túlillesztett lesz a modell. A normált gyakoriságok különbség mátrixait a 5-19. ábra szemlélteti.

Normál adatszeperáció (összeg=0,533)

Kiegyensúlyozott adatszeperáció (összeg=0,373)



5-19. ábra: Normált gyakoriságok különbség mátrixai a két adatszeperációs eljárás esetén

A fenti mátrixok alapján még nem látszik pontosan jól, hogy vajon mennyiben lett kiegyensúlyozottabb a tanító és teszhalmaz, viszont, ha az egyes mátrixok elemét abszolút értékben összegezzük, akkor a kisebb értékhez fog tartozni a kiegyensúlyozottabb adathalmaz. A kiegyensúlyozás esetében sikeresnek volt mondható, mivel a hibaösszeget nagyjából 30%-kal csökkentettem. Az eredményeket a 5-19. ábra mutatja. A kiegyensúlyozás mértéke nem jelentős a rendelkezésemre álló kis elemszámú adatsoron, azonban több ezres nagyságrendű minta esetén komoly különbséget jelenthetne. A következő fejezetben a 5.4 alfejezetben tárgyalt architektúrák viselkedését vizsgáltam a kiegyensúlyozott adatszeperációval létrehozott tanító és teszhalmazon.

5.6. A vizsgált architektúrák viselkedése a kiegyensúlyozott adatsoron

Az egyes architektúrák kiegyensúlyozott adatsoron való eredményességét a 5.4 fejezetben közölt eredményekhez képest, táblázatos formában adtam meg, mivel egyrészt a különbségek nem annyira számottevőek, hogy az ábrákon látszódjanak, másrészt pedig ilyen kis elemszámú adatsor esetén maximum a tendenciák figyelhetőek meg az eredmények változásain keresztül.

5.6.1. EGY REJTETT RÉTEGŰ ARCHITEKTÚRÁK VISELKEDÉSE A KIEGYENSÚLYOZOTT ADATSORON

5.6.1.1. ReLU aktivációs függvénnyel ellátott architektúrák viselkedése

Az egy rejtett réteggel, ReLU aktivációs függvénnyel rendelkező architektúrák eredményességének változását a 5.4.1.1 alfejezetben foglaltakhoz képest százalékosan a 5-6. táblázat foglalja össze.

5-6. táblázat: Egy rejtett rétegű, ReLU aktivációs függvényt használó modellek jellemzőinek változása

	f _{r1}	n _{r1}	f _{r2}	n _{r2}	f _k	p	R ²			MSE _{átlag}	i _{átlag}	div
							R ² _{átlag}	σ _{R2}	MAX(R ²)			
1	ReLU	16	-	-	ReLU	257	-2,3%	-14,7%	-3,6%	-2,0%	0,0%	-57,1%
2	ReLU	32	-	-	ReLU	513	-2,4%	-11,8%	4,3%	-21,2%	0,1%	-33,3%
3	ReLU	64	-	-	ReLU	1025	15,4%	-18,8%	8,5%	-45,2%	-2,0%	0,0%

A fenti táblázat alapján látható, hogy a becslés átlagos és maximális pontossága nem változott számottevően, míg az eredmények szórása szisztematikus csökkenést mutatott. A 16 és 32 neuront tartalmazó architektúrák jelentősen kevesebbszer divergáltak.

5.6.1.2. Tanh aktivációs függvénnyel ellátott architektúrák viselkedése

Az egy rejtett réteggel, *tanh* aktivációs függvénnyel rendelkező architektúrák eredményességének változását a 5.4.1.2 alfejezetben foglaltakhoz képest százalékosan a 5-7. táblázat foglalja össze.

5-7. táblázat: Egy rejtett rétegű, tanh aktivációs függvényt használó modellek jellemzőinek változása

	f _{r1}	n _{r1}	f _{r2}	n _{r2}	f _k	p	R ²			MSE _{átlag}	i _{átlag}	div
							R ² _{átlag}	σ _{R2}	MAX(R ²)			
4	tanh	16	-	-	ReLU	257	0,3%	28,4%	0,4%	-8,0%	0,0%	0,0%
5	tanh	32	-	-	ReLU	513	-0,4%	1,4%	-0,7%	-1,0%	-1,1%	0,0%
6	tanh	64	-	-	ReLU	1025	-1,8%	182,7%	-0,5%	38,1%	-4,8%	0,0%

A táblázat alapján megállapítható, hogy az egy rejtett réteggel, *tanh* aktivációs függvénnyel rendelkező modell a kiegyensúlyozott adatsoron való futtatások után minden eredményességi mutatójában romlott.

5.6.2. KÉT REJTETT RÉTEGŰ ARCHITEKTÚRÁK VISELKEDÉSE A KIEGYENSÚLYOZOTT ADATSORON

5.6.2.1. ReLU aktivációs függvénnyel ellátott architektúrák viselkedése

A két rejtett réteggel, ReLU aktivációs függvénnyel rendelkező architektúrák eredményességének változását a 5.4.2.1 alfejezetben foglaltakhoz képest százalékosan a 5-8. táblázat foglalja össze.

5-8. táblázat: Két rejtett rétegű, ReLU aktivációs függvényt használó modellek jellemzőinek változása

	f _{r1}	n _{r1}	f _{r2}	n _{r2}	f _k	p	R ²			MSE _{átlag}	i _{átlag}	div
							R ² _{átlag}	σ _{R2}	MAX(R ²)			
7	ReLU	32	ReLU	8	ReLU	721	4,7%	-36,6%	1,1%	-31,2%	-5,2%	-50,0%

8	ReLU	32	ReLU	16	ReLU	993	1,9%	-13,0%	1,8%	-19,2%	-4,5%	-55,6%
9	ReLU	32	ReLU	32	ReLU	1537	0,7%	14,3%	0,1%	-13,7%	-7,8%	-38,5%

Az eredmények azt bizonyítják, hogy a két rejtett réteggel, kizárólag *ReLU* aktivációs függvényekkel rendelkező modellek a kiegyensúlyozott adatsoron átlagos és legjobb pontosságukat, valamint tanulási gyorsaságukat tekintve kissé, míg a konvergenciájukhoz kötődő tulajdonságaikban jelentősen javultak. A 32-32 neuront hordozó modell eredményeinek szórása azonban nőtt.

5.6.2.2. *Tanh* aktivációs függvénnyel ellátott architektúrák viselkedése

A két rejtett réteggel, *tanh* aktivációs függvénnyel rendelkező architektúrák eredményességének változását a 5.4.2.2 alfejezetben foglaltakhoz képest százalékosan a 5-9. táblázat foglalja össze.

5-9. táblázat: Két rejtett rétegű, *tanh* aktivációs függvényt használó modellek jellemzőinek változása

	f_{r1}	n_{r1}	f_{r2}	n_{r2}	f_k	p	R^2			$MSE_{\text{átlag}}$	$\bar{a}_{\text{tlag}}$	div
							$R^2_{\text{átlag}}$	σ_{R^2}	$MAX(R^2)$			
10	tanh	32	ReLU	8	ReLU	721	-1,3%	0,4%	1,1%	-14,2%	0,7%	-60,0%
11	tanh	32	ReLU	16	ReLU	993	-3,4%	-10,5%	0,6%	-11,8%	1,6%	0,0%

A táblázat alapján az előző fejezethez hasonlóan az a következtetés vonható le, hogy a modell konvergenciáját jelentősen javítja a tanító és teszt halmazok kiegyensúlyozása. A modell pontossága és tanulási gyorsasága kis mértékben romlott.

A fejezet egészét tekintve nagy biztonsággal kijelenthető, hogy a 5.4-es fejezetben robusztusként aposztrofált modellek esetén nem érdemes az adatsor általam javasolt kiegyensúlyozását alkalmazni, viszont a kevésbé robusztus modellek kiegyensúlyozott adatsoron való tanítása a divergens esetek számát jelentősen csökkenti.

5.7. A bemeneti változók csökkentése

A teljesítmény, illetve a belépő hűtővíz hőmérséklet mindig a bemeneti adatsor részét képezi a hatásfok becslése során, hiszen ezen két változó teljesen független egymástól, illetve az erőmű üzeme során ezek a külső körülmények miatt adódnak. Az általam modellezett 800 MW névleges teljesítményű szuperkritikus szénerőmű feltehetőleg részt vesz az aFRR (automatic frequency restoration reserve) vagy primer szabályozásban, vagyis a létesítmény teljesítményét a villamosenergia-rendszer adott pillanatbeli állapota szabja meg. A kondenzátorba belépő hűtővíz hőmérséklete pedig egyértelműen a külső környezeti hőmérséklettől függ. Azt a bemeneti adatsort, melylyel a lehető legjobb hatásfokbecslés kapható tehát a 12 üzemi paraméter (T_{tv} , m_{tv} , p_{uh} , m_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{E7} , p_{gtt}) valamely kombinációja a teljesítménnyel és hűtővíz hőmérséklettel kiegészítve fogja kiadni.

Ahhoz, hogy a legjobb kombináció meghatározása lehetségessé váljon olyan hálóarchitektúrát kellett választanom, melynek szórása a lehető legkisebb. Erre azért volt szükség, hogy egyértelműen el lehessen különíteni a determinációs együttható változásának okát, vagyis, hogy a modell becslőkéességének változása az eltérő magyarázó változóknak vagy a véletlennek volt köszönhető. A legkisebb szórással az egy rejtett rétegű, 64 neuront és $\tanh$ aktivációs függvényt tartalmazó háló rendelkezik, így a vizsgálataimat ezen modell használatával folytattam le.

Munkám során a kimerítő keresés módszerét alkalmaztam, azaz a bemeneti adatsorokat a 12 üzemi paraméter 11 esetre vett különböző ismétlés nélküli kombinációi alapján képeztem. A magyarázó változók közötti összes kereszthatás feltérképezéséhez végig kellett iterálni az összes olyan eseten, mely során a 12 üzemi paraméterből 1-et, 2-öt, 3-at, ..., 11-et hagyunk el, így a képzett adatsorok a teljesítménnyel és hűtővíz hőmérséklettel kiegészülve maximum 14, az egyes üzemi paraméterek elhagyásával 13, 12, ..., 3 változóból álltak. Az ezen megközelítés alapján képezhető különböző bemeneti adatsorok száma 4094 a (32) összefüggés szerint.

$$\sum_{i=1}^{11} \binom{12}{i} = 4094 \quad (32)$$

Az eredmények megbízhatóságának növelése érdekében a neurális hálót 100 alkalommal tanítottam be és értékeltem ki, a tanítási ciklusok száma 250 volt. A Python „joblib” könyvtárának segítségével párhuzamosítottam az algoritmus futását, ezzel jelentősen rövidült a futáshoz szükséges idő. A processzor logikai számítási egységeinek megfelelő számú párhuzamos szálon voltam képes futtatni a kódot. Az egyes szaklak a 100 tanítási és kiértékelési ciklus közül egyet-egyet számítottak egyidejűleg.

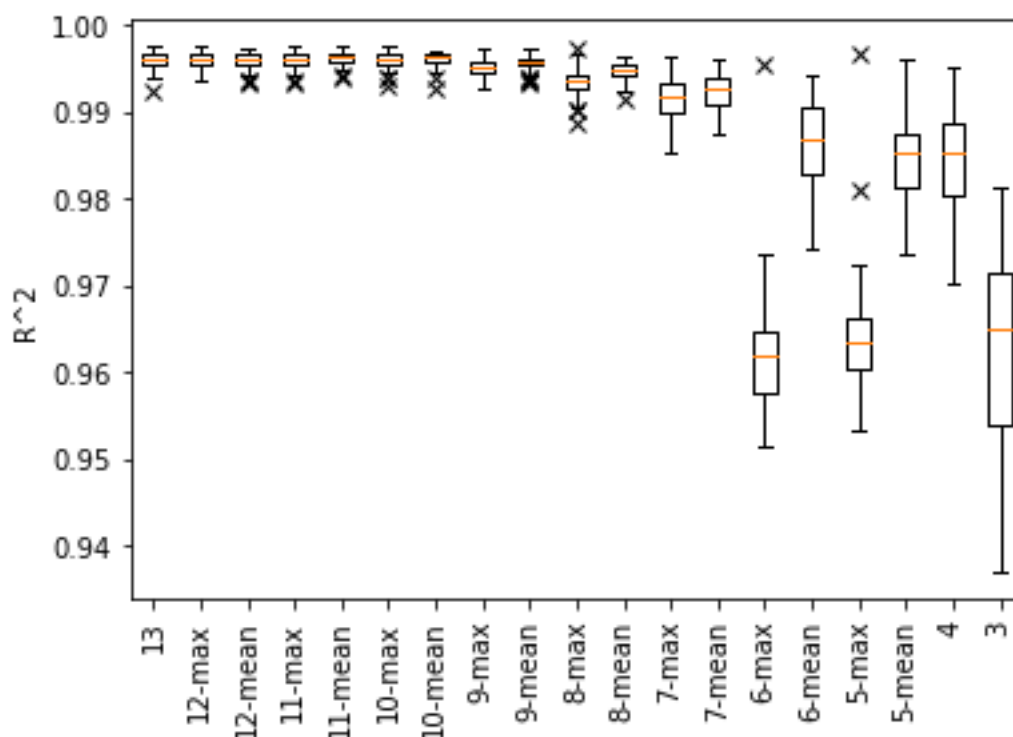
Az algoritmus tesztelése során már látszott, hogy a saját számítógépemen ilyen nagy mennyiségű iteráció lefutása több napot venne igénybe még a párhuzamosítással együtt is. A saját gépem 1 db 1,6 GHz-es, 4 magos, 8 logikai számítási egységgel rendelkező Intel Core i5-8265U processzorral van ellátva. Az Energetikai Gépek és Rendszerek Tanszék ezért rendelkezésemre bocsátotta az egyik szerverét, mellyel a futási idő nagyjából a hetedére csökkent. A szerver 2 db 1,2 GHz-es, 16 magos, 32 logikai számítási egységgel rendelkező AMD EPYC 7301 processzort tartalmaz, így 64 párhuzamos szálon voltam képes futtatni az algoritmust.

A saját laptopomra becsült futási időket a legjobb 13 változó meghatározásához tartozó kód futási idejének lineáris extrapolációjával kaptam. A szerverre előzetesen számolt futási idők ennek nyolcadával lettek becsülve, mivel a tanszéki gépen nyolcszor annyi párhuzamos szálon tudtam futtatni a modellt. A becsült és valós futási időket a 5-10. táblázat tartalmazza. A táblázat alapján látható, hogy a bemeneti adatsor összes kombinációját vizsgáló algoritmus nagyjából 21 óra alatt futott le, ez az idő a saját rendszeremen több, mint 145 órát vett volna igénybe.

5-10. táblázat: Becsült és valós futási idők a két rendszerre

Változók száma	Kombinációk száma	Becsült futási idő (8 szálon) [min]	Becsült futási idő (64 szálon) [min]	Valós futási idő (64 szálon) [min]
13	12	25,6	3,2	4,4
12	66	140,8	17,6	23,3
11	220	469,3	58,7	76,6
10	495	1056,0	132,0	163,5
9	792	1689,6	211,2	257,6
8	924	1971,2	246,4	276,3
7	792	1689,6	211,2	236,2
6	495	1056,0	132,0	138,3
5	220	469,3	58,7	58,9
4	66	140,8	17,6	16,1
3	12	25,6	3,2	3,0
Összesen	4094	8733,9	1091,7	1254,2

A vizsgálat végén 11 db táblázatot kaptam, melyekben az oszlopok száma a kombinációk számával, a sorok száma 100-zal egyezett meg, azaz minden kombinációhoz 100 determinációs együttható került kiszámításra, hogy a modellek szórása kellően figyelembe legyen véve. Az adatok kiértékelésekor oszloponként definiáltam a determinációs együtthatók átlagát és maximum értékét. Ezután megvizsgáltam, hogy mely kombinációhoz tartozik a legnagyobb átlagos és maximális R^2 érték. Ha mindkét esetben ugyanaz a kombináció adódott legideálisabbként, akkor egyértelműen meghatároztam, hogy a vizsgált változó szám esetén, mely kombinációval adódik az optimális becslés. Ez csak a legjobb 3, 4, valamint 13 változóhoz tartozó esetekben volt igaz, a többi esetben két potenciális legjobb kombinációt definiáltam. Az előbb említett kombinációkhoz tartozó 100 determinációs együtthatót „boxplot” segítségével jelenítettem meg, mely szemlélteti az egyes adatsorok eloszlását, maximum, minimum, illetve átlag értékeit. A kiugró értékeket „x”-ek jelzik. Ezt a 5-20. ábra mutatja.



5-20. ábra: Legjobb kombinációkhoz tartozó determinációs együtthatók

A fenti ábra alapján belátható, hogy a 100 futtatás alatt legjobb determinációs együtthatókat produkáló modellek összességében nagy szórásuk miatt kevésbé voltak robusztusak, mint amelyek az átlagos determinációs együtthatók alapján lettek definiálva. Ezért a legkedvezőbb becslést biztosító bemeneti adatsorok meghatározását csak a legjobb R^2 átlagok alapján végeztem el. A legjobb kombinációkat a 5-11. táblázat tartalmazza. A maximális R^2 -ek alapján definiált legjobb kombinációk táblázatát a mellékletek között, a 8. fejezetben közöltem.

5-11. táblázat: Legjobb kombinációk a változók száma szerint (R^2 átlag alapján)

Változók száma	Paraméterek száma	Legjobb kombináció (átlag)	$R^2_{\text{átlag}}$
3	321	P, T_{hvbe} , T_{tv}	0,9625
4	385	P, T_{hvbe} , T_{tv} , p_{E1}	0,9843
5	449	P, T_{hvbe} , T_{tv} , p_{E1} , p_{E2}	0,9846
6	513	P, T_{hvbe} , m_{kond} , p_{E1} , p_{E3} , p_{E4}	0,9866
7	577	P, T_{hvbe} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4}	0,9924

8	641	$P, T_{hvbe}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}$	0,9948
9	705	$P, T_{hvbe}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}, p_{gtt}$	0,9957
10	769	$P, T_{hvbe}, p_{uh}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}, p_{gtt}$	0,9961
11	833	$P, T_{hvbe}, p_{uh}, m_{uh}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}, p_{gtt}$	0,9963
12	897	$P, T_{hvbe}, p_{uh}, m_{uh}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}, p_{E7}, p_{gtt}$	0,9960
13	961	$P, T_{hvbe}, m_{tv}, p_{uh}, m_{uh}, m_{kond}, p_{E1}, p_{E2}, p_{E3}, p_{E4}, p_{E5}, p_{E7}, p_{gtt}$	0,9960

A legpontosabb átlagos nettó hatásfok predikció 11 magyarázó változóval állt elő 833 modellparaméter mellett. A **determinációs együttható** jelentős javulást mutatott a 14 változóval végzett számításokhoz képest, hiszen **0,938-as értékről 0,9963-ra nőtt**, a modellparaméterek száma pedig 1025-ről 833-ra csökkent. A legpontosabb becslés a következő üzemi paraméterek alapján jött létre:

- Teljesítmény (P)
- Kondenzátorba belépő hűtővíz hőmérséklete (T_{hvbe})
- Újrahevítési nyomás és tömegáram (p_{uh}, m_{uh})
- Kondenzátorba belépő munkaközeg tömegárama (m_{kond})
- 1-5. számú tápvíz-előmelegítők megcsapolási nyomásai ($p_{E1}, \dots, p_{E5}$)
- Gáztalanító táptartály fűtőgőzének nyomása (p_{gtt})

A következőkben termodinamikai megfontolások alapján értékeltem a 5-11. táblázatban foglalt eredményeket.

A teljesítmény és hűtővíz hőmérséklet után elsőként a tápvíz hőmérséklet (T_{tv}) jelent meg a paraméterek között. Ez termodinamikai megfontolások alapján sem meglepő, hiszen a tápvíz hőmérséklet változása közvetlen hatással van a bevezetett hőteljesítményre.

A második és harmadik üzemi paraméter, mellyel a legjobb becslés adódott, a kondenzátorhoz két legközelebbi tápvíz-előmelegítő fokozat megcsapolási nyomása (p_{E1} ,

p_{E2}) lett. A négy legjobb leírást biztosító üzemi paraméter meghatározása esetén furcsa dolog történt, mivel a tápvíz hőmérséklet, valamint a második tápvíz-előmelegítő fokozat megcsapolási nyomása kiesett a paraméterek közül, és ezeket felváltotta a kondenzátorba belépő gőz tömegárama (m_{kond}), valamint a harmadik és negyedik fokozathoz tartozó megcsapolási nyomás (p_{E3} , p_{E4}). Ez első közelítésben ellentmondásnak tűnhet, azonban a tápvíz-előmelegítők hőmérlegeiben használt megcsapolási entalpiák nyomástól is függenek, így a fokozatok közötti hőmérsékletek és a tápvíz hőmérséklet is a megcsapolási nyomások függvénye. A kondenzátorba belépő, vagyis a gőzturbinából kilépő munkaközeg tömegárama szintén kapcsolatba hozható a tápvíz-előmelegítő rendszerrel (TVER) oly módon, hogy a tápvíz tömegáramból kivonjuk az összes megcsapolás tömegáramát. A modell tehát ugyanazt a hatást több változóval pontosabban írta le.

A hat és kilenc magyarázó változós esetek között nagyjából az látszik, hogy a modell a kondenzátorba belépő gőztömegáramot és a kis nyomású TVER megcsapolási nyomásait vette figyelembe, általában az eggyel nagyobb nyomású elemmel kiegészítve a bemeneti adatsort. Ez nem egészen igaz például a nyolc változós esetben mivel ott p_{E4} helyett a gáztalanítás táptartály fűtőgőz nyomásának beemelésével volt a legpontosabb a becslés. A tendencia megtörésének oka a modell szórásának is köszönhető. Ezek után a modell az első nagynyomású TVE fokozat megcsapolási nyomását (p_{E5}) emelte be, majd rend szerint az újrahevítési nyomást és tömegáramot (p_{uh} , m_{uh}). Az utolsó két lépésben a bemeneti adatsor a nagynyomású TVER utolsó fokozati megcsapolási nyomásával (p_{E7}) és a tápvíz tömegárammal (m_{tv}) egészült ki.

A fentiek alapján kijelenthető, hogy a nagy mennyiségű számítással meghatározott legjobb üzemi paraméterek megjelenése termodinamikailag is könnyen belátható.

6. ÖSSZEFOGLALÁS

Dolgozatomban a neurális hálózatok erőműiparban való alkalmazását tárgyaltam, megvalósított ipari példákon keresztül, kitérve a stacioner és tranziens üzemállapotok modellezésének lehetőségére, valamint diagnosztikai felhasználásokra. Bemutattam a hálók biológiai asszociációs hátterét, az attól való leglényegesebb eltéréseket. Részletesen leírtam a feladatom tárgyául szolgáló mesterséges intelligencián alapuló módszer tanulási és számítási algoritmusai mögött húzódó matematikai apparátust.

Egy saját magam által CycleTempo szoftverben modellezett 800 MW-os szuperkritikus szénerőművi blokk nettó hatásfokát neurális háló segítségével prognosztizáltam. A tanításhoz szükséges adatsort a CycleTempo-ból kinyert nyers adatok, illetve Python-ban létrehozott jelleggörbék alapján képeztem. Függvénykönyvtárak segítségével összesen 11 különböző hálóarchitektúrát hoztam létre. A hálók 250 tanítási ciklussal futottak, illetve 100 alkalommal lettek újratanítva és kiértékelve. Mivel a modellparaméterek kezdeti értékei minden alkalommal más értéken lettek inicializálva, a kapott eredmények szórtak. Az egyes modellek által nyújtott regresszió pontosságát, a tanulás gyorsaságát és konvergenciáját átfogóan, statisztikai mutatók és diagramok segítségével elemeztem.

A vizsgált modellek közül három architektúra biztosított elfogadható pontosságú becslést, megfelelő konvergencia és szórás mellett. A leggyorsabb és átlagosan legpontosabb modellnek a két rejtett rétegű, 32-32 neuront, illetve *ReLU* aktivációs függvényeket tartalmazó háló bizonyult. Robosztusság szempontjából ezt felülmúlták az egy rejtett rétegű *tanh* aktivációs függvényt alkalmazó 32 és 64 neuront tartalmazó hálók. Ezek szórása kisebb volt, pontosságuk pedig nem sokkal maradt el a két rejtett rétegű hálótól.

Munkám során értékeltem, hogy a tanító és teszt adatsorok közötti egyenletesebb adateloszlás esetén hogyan változnak az eredményességi mutatók az egyes hálókra nézve. Az adatsor finomabb felosztása nem volt különösen jó hatással a kis szórású modellekre, viszont a kevésbé robusztus modellek konvergenciáját nagyban sikerült javítanom a módszerrel.

Végül a kimerítő keresés módszerét alkalmazva meghatároztam, hogy mely üzemi paraméterkombinációból álló adatsorral szolgáltatja a legjobb hatásfokbecslést a modell. Az algoritmus az egy rejtett rétegű *tanh* aktivációs függvényt tartalmazó architektúrával futott. Az iterációk számossága miatt számításaimat az Energetikai Gépek és Rendszerek Tanszék egyik szerverén futtattam. A legjobb hatásfokbecslést 11 magyarázó változóval értem el, mely modell 833 paramétert tartalmazott. A determinációs együttható ebben az esetben 0,9963-ra adódott. Ehhez képest R^2 értéke, amikor a modell mind a 14 változó alapján tanult csupán 0,938 volt 1025 modellparaméter mellett.

A legjobb átlagos determinációs együtthatót produkáló kombinációkat 3, 4, ..., 13 db változóra is meghatároztam. Az eredmények összeegyeztethetőek voltak a termodinamikai megfontolásokkal. A bemeneti változók csökkentésére vonatkozó részletes eredmények az 5.7 fejezetben olvashatók.

A fenti eredmények és módszerek alkalmazásával olyan neurális háló konstruálható, amely nagy pontossággal képes kiszámítani egy hőerőmű nettó hatásfokát annak üzemi paraméterei alapján.

7. FELHASZNÁLT FORRÁSOK

- [1] Power Online: „GE Power Systems selects GateCycle”, 2001.03.05. [Online]. Elérhető: <https://www.poweronline.com/doc/ge-power-systems-selects-gatecycle-0001>. [Hozzáférés dátuma: 2022.02.20.]
- [2] Steag Energy Services: „Design and engineer your plants with EBSILON® Professional – STEAG’s innovative, world-leading planning tool”, é.n. [Online]. Elérhető: <https://www.ebsilon.com/en/>. [Hozzáférés dátuma: 2022.02.20.]
- [3] W. Chen, Y. Liang, X. Luo, J. Chen, Z. Yang, Y. Chen (2021): Artificial neural network grey-box model for design and optimization of 50 MWe-scale combined supercritical CO₂ Brayton cycle-ORC coal-fired power plant. Elsevier, Energy Conversion and Management 249. kötet. DOI: <https://doi.org/10.1016/j.enconman.2021.114821>.
- [4] F. Rossi, D. Velázquez, I. Monedero, F. Biscarri (2014): Artificial neural networks and physical modeling for determination of baseline consumption of CHP plants, Elsevier, Expert Systems with Applications 41. kötet, pp. 4658-4669. DOI: <https://doi.org/10.1016/j.eswa.2014.02.001>.
- [5] S. Dea, M. Kaiadi, M. Fast, M. Assadi (2007): Development of an artificial neural network model for the steam process of a coal biomass cofired combined heat and power (CHP) plant in Sweden, Elsevier, Energy 32. kötet, pp. 2099-2109. DOI: 10.1016/j.energy.2007.04.008.
- [6] Michael Nielsen (2019): Neural Networks and Deep Learning, [Online]. Elérhető: <http://neuralnetworksanddeeplearning.com/index.html>. [Hozzáférés dátuma: 2022.02.21.]
- [7] Császár Márk (2017): „Mély neurális hálózatok alkalmazása pénzügyi idősorok modellezésére historikus adatok és szövegelemzés alapján”, Budapesti Műszaki és Gazdaságtudományi Egyetem.
- [8] Rizwan Alam: „Normalization vs Standardization Explained” 2020.05.05. [Online]. Elérhető: <https://towardsdatascience.com/normalization-vs-standardization-explained-209e84d0f81e>. [Hozzáférés dátuma: 2022.02.22.]
- [9] Analytics Vidhya: „CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning”, 2020.02.17. [Online]. Elérhető: <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/>. [Hozzáférés dátuma: 2022.03.01.]
- [10] Ryno Laubscher (2019): Time-series forecasting of coal-fired power plant reheater metal temperatures using encoder-decoder recurrent neural networks, Elsevier, Energy 189. kötet. DOI: <https://doi.org/10.1016/j.energy.2019.116187>.
- [11] G. Lee, S. J. Lee, C. Lee(2021): A convolutional neural network model for abnormality diagnosis in a nuclear power plant, Elsevier, Applied Soft Computing Journal 99. kötet. DOI: <https://doi.org/10.1016/j.asoc.2020.106874>.

- [12] Benyó Balázs, BME: Mesterséges neurális hálózatok [Online]. Elérhető: https://semmelweis.hu/kepalkotas/files/ANN_BenyoB.pdf. [Hozzáférés dátuma: 2022. 05. 01.]
- [13] MIT Open Course Ware: Lecture 12A Neural Nets [Online]. Elérhető: <https://ocw.mit.edu/courses/6-034-artificial-intelligence-fall-2010/resources/lecture-12a-neural-nets/>. [Hozzáférés dátuma: 2022. 05. 01.]
- [14] Serhat Seker, Emine Ayaz, Erdinc Türkcán (2003): Elman's recurrent neural network applications to condition monitoring in nuclear power plant and rotating machinery, Elsevier, Engineering Applications of Artificial Intelligence 16. kötet 647-656 oldal. DOI: 10.1016/j.engappai.2003.10.004
- [15] MIT Introduction to Deep Learning 6.S191: Lecture 2: Recurrent Neural Networks, 2022. január [Online]. Elérhető: https://www.youtube.com/watch?v=QvkQ1B3FBqA&ab_channel=AlexanderAmini. [Hozzáférés dátuma: 2022. 05. 02.]
- [16] MIT Introduction to Deep Learning 6.S191: Lecture 3: Convolutional Neural Networks for Computer Vision, 2022. január [Online]. Elérhető: https://www.youtube.com/watch?v=uapdILWYTzE&ab_channel=AlexanderAmini. [Hozzáférés dátuma: 2022. 05. 02.]
- [17] M.V.J.J. Suresh, K.S. Reddy, Ajit Kumar Kolar (2011): ANN-GA based optimization of a high ash coal-fired supercritical power plant, Elsevier, Applied Energy 88. kötet, pp. 4867-4873. DOI.: 10.1016/j.apenergy.2011.06.029
- [18] Pintér József, Rappai Gábor (2007): Statisztika
- [19] Semmelweis Egyetem: Regressziós vizsgálatok [Online]. Elérhető: https://semmelweis.hu/kutlab/files/2019/03/regresz-szi%3b3_el%c5%91ad%c3%a1s_DE_2019.pdf. [Hozzáférés dátuma: 2022. 09. 07.]
- [20] Robert May, Graeme Dandy and Holger Maier (2011). Review of Input Variable Selection Methods for Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications, Prof. Kenji Suzuki (Szerk.), ISBN: 978-953-307-243-2, InTech, Elérhető: <http://www.intechopen.com/books/artificialneural-networks-methodological-advances-and-biomedical-applications/review-of-input-variable-selectionmethods-for-artificial-neural-networks>

8. MELLÉKLET

8-1. táblázat: Legjobb kombinációk a változók száma szerint (R^2 maximuma alapján)

Változók száma	Paraméterek száma	Legjobb kombináció (max)	$R^2_{\max}$
3	321	P, T_{hvbe} , T_{tv}	0,9812
4	385	P, T_{hvbe} , T_{tv} , p_{E1}	0,9953
5	449	P, T_{hvbe} , T_{tv} , p_{E4} , p_{E7}	0,9968
6	513	P, T_{hvbe} , T_{tv} , m_{kond} , p_{E4} , p_{E7}	0,9954
7	577	P, T_{hvbe} , m_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3}	0,9965
8	641	P, T_{hvbe} , m_{tv} , p_{uh} , m_{kond} , p_{E1} , p_{E3} , p_{gtt}	0,9972
9	705	P, T_{hvbe} , p_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E7} , p_{gtt}	0,9974
10	769	P, T_{hvbe} , m_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{gtt}	0,9975
11	833	P, T_{hvbe} , m_{tv} , p_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{gtt}	0,9978
12	897	P, T_{hvbe} , m_{tv} , p_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{E7} , p_{gtt}	0,9977
13	961	P, T_{hvbe} , m_{tv} , p_{uh} , m_{uh} , m_{kond} , p_{E1} , p_{E2} , p_{E3} , p_{E4} , p_{E5} , p_{E7} , p_{gtt}	0,9977