ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԿԻՐԱՌԱԿԱՆ ՄԱԹԵՄԱՏԻԿԱՅԻ ՖԱԿՈՒԼՏԵՏ

ԾՐԱԳՐԱՎՈՐՄԱՆ ԵՎ ԻՆՖՈՐՄԱՑԻՈՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐԻ

ԱՄԲԻՈՆ

# Ավարտական աշխատանք

Թեմա՝ Խոսքից լեզվի ճանաչում խորը ուսուցման մեթոդներով

Ուսանող՝ Հրայր Հարությունյան

Ղեկավար՝ Արմեն Անդրեասյան

ԵՐԵՎԱՆ 2016

# Բովանդակություն

# 1 Ներածություն

Խոսքից լեզվի ճանաչման խնդիրը տրված խոսքի ձայնագրության մեջ հնչող լեզվի որոշումն է: Մարդը այս խնդիրը կարող է լուծել անգամ առանց տվյալ լեզվի իմացության: Համակարգչի համար պատկերը այլ է. համակարգչում խոսքի ձայնագրությունը ներկայացվում է որպես թվերի հաջորդականություն, որի հիման վրա անհրաժեշտ է որոշել լեզուն: Սահմանված խնդիրը տվյալների դասակարգման (data classification) խնդիր է, և նրա համար կիրառելի են մեքենայական ուսուցման (machine learning) բազմաթիվ մեթոդներ: Այդ մեթոդների մի մասում անհրաժեշտ է մասնագետի միջամտություն: Մասնագետը առանձնացնում է ձայնագրության այնպիսի հատկանիշներ (feature extraction), որոնք նա կարծում է, որ կարող են օգտակար լինել խնդրի լուծման համար: Այդ հատկանիշների վրա էլ կիրառվում է դասակարգում իրականացնող մեքենայական ուսուցման ինչ-որ մեթոդ:

Վերջին տասնամյակում հաշվողական տեխնիկայի հզորության մեծացումը նպաստեց խորը ուսուցման (deep learning) մեթոդների զարգացմանը, որոնք գրեթե չեն պահանջում մասնագետի միջամտություն և կարողանում են տվյալների համար ստանալ այնպիսի ներկայացում (data representation), որի հիման վրա մեքենայական ուսուցման ստանդարտ մեթոդները հասնում են բավականաչափ լավ արդյունքների: Օրինակ, վերջին տարիներին խորը փաթույթային նեյրոնային ցանցերի (deep convolutional neural networks) օգնությամբ ռեկորդային արդյունքներ են գրանցվում նկարներից օբյեկտների ճանաչման և խոսքի ճանաչման բազմաթիվ խնդիրներում: Խորը նեյրոնային ցանցի միջոցով 2012թ-ին հաջողվեց մեծ առավելությամբ հաղթել [5] նկարներից օբյեկտների ճանաչման ImageNet Large Scale Visual Recognition Competition (ILSVRC) մրցույթը, որում մինչև այդ առաջատար էին մասնագետի միջամտությամբ աշխատող մեթոդները:

## 1.1 Տվյալների ներկայացում

Մեքենայական ուսուցման մեջ առանցքային խնդիր է տվյալների նոր ներկայացում գտնելը: Նոր ներկայացում գտնելու նպատակներից են՝

- Տվյալների սեղմում
- Աղմուկի հեռացում

- Ավելի «լավ» ներկայացման որոնում

Ավելի «լավ» ներկայացումը ենթադրում է տվյալների այնպիսի ներկայացում, որի վրա մեքենայական ուսուցման մեթոդների կիրառումը ավելի արդյունավետ է: Սովորաբար ենթադրվում է, որ նոր ներկայացումը պետք է պարունակի տվյալները նկարագրող կարևոր հատկանիշներ և չպարունակի աղմուկ:

Տվյալների նոր ներկայացում կարելի է գտնել բազմամակարդակ, հիերարխիկ մոտեցումով՝ ունենալով տվյալների սկզբնական ներկայացումը, ամեն քայլում նախորդ ներկայացման հիման վրա կառուցվում է նոր ներկայացում:

## 1.2 Խորը ուսուցում և խորը նեյրոնային ցանցեր

Խորը ուսուցումը մեքենայական ուսուցման ճյուղ է, որն ուսումնասիրում է խորը ներկայացում գտնող մեթոդներ: Խորը ուսուցման մոդելների ընտանիք են նեյրոնային ցանցերը: Նեյրոնային ցանցերը մեքենայական ուսուցման մոդելներ են, որոնք նման են կենսաբանական նեյրոնային ցանցերին և օգտագործվում են տարբեր տիպի ֆունկցիաների մոտարկման համար: Նեյրոնային ցանցը կազմված է բազմաթիվ նեյրոններից, որոնց միջև կան միակողմանի կապեր: Այդ կապերի միջոցով մի նեյրոնից ինֆորմացիան փոխանցվում է մեկ այլ նեյրոնի: Յուրաքանչյուր նեյրոն ունի մուտք, ելք և ակտիվացիայի ֆունկցիա: Մուտքում ստանալով ինչ-որ թվեր, նեյրոնը կազմում է այդ թվերի գծային կոմբինացիա (այդ կոմբինացիայի գործակիցներին կանվանենք նեյրոնի պարամետրեր), որի վրա կիրառում է իր ակտիվացիայի ֆունկցիան: Ստացվածը կոչվում է նեյրոնի ակտիվացիա, որը հանդիսանում է նեյրոնի ելքը և հետագայում կարող է փոխանցվել այլ նեյրոնի: Ցանցում առկա բոլոր նեյրոնների պարամետրերի բազմությանը կանվանենք ցանցի պարամետրեր: Նեյրոնները իրենց միջև եղած կապերի հետ միասին կազմում են ուղղորդված գրաֆ: Ցանցի այն նեյրոնները, որոնք ոչ մի նեյրոնից ինֆորմացիա չեն ստանում կանվանենք ցանցի մուտքային նեյրոններ, իսկ դրանց բազմությունը՝ ցանցի մուտք: Մուտքային նեյրոնների դեպքում նեյրոնի ակտիվացիա բառակապակցության փոխարեն կօգտագործենք կանվանենք նեյրոնի արժեք բառակապակցությունը: Ցանցում նաև առանձնացվում է նեյրոնների ինչ-որ ենթաբազմություն, որին կանվանենք ցանցի ելք:

Եթե նեյրոնային ցանցի գրաֆում գոյություն չունի ուղղորդված ցիկլ, ապա կասենք, որ ցանցը ուղիղ տարածող նեյրոնային ցանց է (feedforward neural network), հակառակ դեպքում կասենք, որ ցանցը ռեկուրենտ նեյրոնային ցանց է (recurrent neural network)։ Ռեկուրենտ նեյրոնային ցանցերում նեյրոնների ակտիվացիաները հաշվարկվում են դիսկրետ ժամանակի յուրաքանչյուր պահի՝ հիմնվելով տվյալ պահի ցանցի մուտքի և նախորդ պահին նեյրոնների ունեցած ակտիվացիաների վրա։

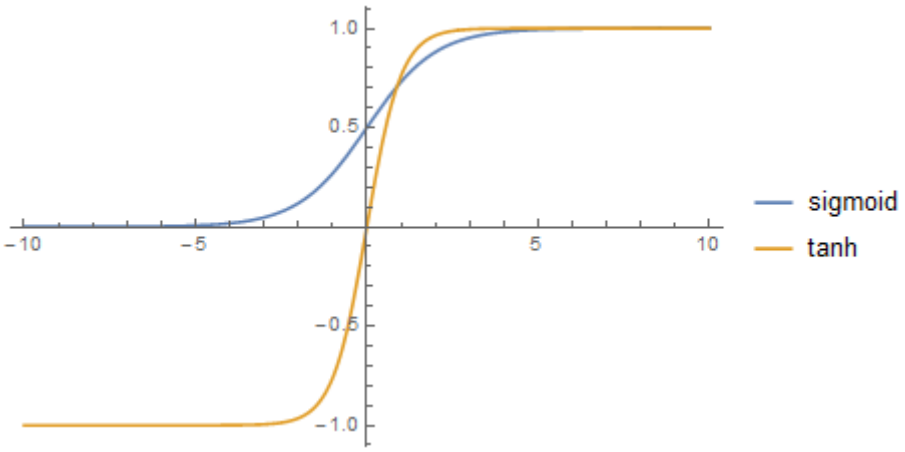
Սովորեցնել նեյրոնային ցանցին մոտարկել ինչ-որ ֆունկցիա նշանակում է ցանցի պարամետրերը ընտրել այնպես, որ երբ ցանցի մուտքային նեյրոնների արժեքներին վերագրվեն ֆունկցիայի արգումենտները, հաշվարկներից հետո ցանցի ելքում մոտարկվի ֆունկցիայի արժեքը այդ արգումենտների դեպքում։ Երբ ցանցը մոտարկում է այնպիսի $f: X \to Y$ ֆունկցիա, որտեղ $Y$-ը վերջավոր բազմություն է, կասենք, որ ցանցը փորձում է լուծել դասակարգման խնդիր։

Նեյրոնային ցանցերը միմյանցից տարբերվում են իրենց կառուցվածքով՝ ինչքան նեյրոն են պարունակում, նեյրոնները ինչ ակտիվացիայի ֆունկցիաներ ունեն, ինչպես են իրար հետ միացված։ Նեյրոնային ցանցի այն հատկանիշները, որոնք չեն փոխվում ուսուցման ընթացքում կոչվում են ցանցի հիպերպարամետրեր։ Ցանցի հիպերպարամետրեր են ցանցի չափերը, ցանցում պարունակվող նեյրոնների քանակը, ցանցին սովորեցնելու ալգորիթմի պարամետրերը և այլն։

### 1.2.1 Ակտիվացիայի ֆունկցիա

Նեյրոնային ցանցերում որպես ակտիվացիայի ֆունկցիա ընտրվում են դիֆերենցելի և մոնոտոն ֆունկցիաներ։ Հաճախ օգտագործվող ֆունկցիաներից են՝

- $id(x) = x$
- $relu(x) = \max(0,\ x)$
- $sigmoid(x) = 1\ /\ (1 + e^{-x})$
- $tanh(x)$
- $softmax(x)_i = \frac{e^{xi}}{\sum_{j=1}^{n} e^{xj}}$




Առաջին 4 ֆունկցիաները մեկ փոփոխականից կախված ֆունկցիաներ են և բազմաչափ զանգվածների վրա կիրառվելուց կիրառվում են անդամ առ անդամ: $Softmax$ ֆունկցիան ստանում է վեկտոր և վերադարձնում վեկտոր: $Softmax$ գործողությունից ստացված վեկտորի բոլոր անդամները դրական թվեր են, իսկ նրանց գումարը հավասար է 1-ի: Այս ակտիվացիայի ֆունկցիան օգտագործվում է դասակարգման խնդիր լուծող նեյրոնային ցանցերում, երբ վերջին շերտում հարկավոր է ստանալ հավանականությունների վեկտոր, որի յուրաքանչյուր բաղադրիչ ցույց կտա ինչ-որ դասի պատկանելու հավանականությունը:

### 1.2.2 Տվյալներ, կորստի ֆունկցիա, էմպիրիկ կորստի ֆունկցիա

Նպատակային $f: X \to Y$ ֆունկցիան, որը պետք է մոտարկենք, սովորաբար տրված է լինում զույգերի հավաքածուով՝ $D = \{(x_i, \quad y_i) \mid i = 1..n,\ y_i = f(x_i)\}$, որոնց կանվանենք տվյալներ: Նեյրոնային ցանցը փորձում է $f$ ֆունկցիայի համար գտնել լավ մոտարկում ֆունկցիաների ինչ-որ դասից: Նեյրոնային ցանցի կողմից $f$ ֆունկցիայի մոտարկումը նշանակենք $\hat{f}$-ով:

$L : Y^2 \to R$ տիպի ֆունկցիային կանվանեք կորստի ֆունկցիա, եթե $L(y, \hat{y})$-ը բնութագրում է $y$-ը $\hat{y}$-ով մոտարկման սխալանքի չափը: Օրինակ $L$ -ը կարող է լինել $Y$ տարածության մեջ հեռավորություն սահմանող կամայական ֆունկցիա: Կորստի

ֆունկցիայի հիման վրա սահմանվում է էմպիրիկ կորստի ֆունկցիա՝ $\tilde{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L\left(y, \hat{f}(x_i)\right)$, որտեղ $\theta$-ն ցանցի բոլոր պարամետրերից ստացված վեկտորն է։

Նեյրոնային ցանցերի նպատակն է գտնել այնպիսի $\theta$ վեկտոր, որի համար էմպիրիկ կորստի ֆունկցիան կլինի հնարավորինս փոքր։ Սովորաբար $\hat{f}$-ը և $\tilde{L}$-ը լինում են դիֆերենցելի, և $\tilde{L}(\theta)$-ն մինիմիզացնելու համար կիրառելի են դառնում մաթեմատիկական օպտիմիզացիայի մեթոդները։

## 1.3 Բազմաշերտ նեյրոնային ցանց

Նեյրոնային ցանցին կանվանենք բազմաշերտ նեյրոնային ցանց, երբ նեյրոնները խմբավորված են հաջորդական շերտերով այնպես, որ նույն շերտի նեյրոնների միջև կապեր չկան, և ամեն նեյրոն կարող է որպես մուտք ընդունել միայն իր շերտին նախորդող շերտի նեյրոնների ակտիվացիաները։ Առաջին շերտը կլինի ցանցի մուտքը, իսկ վերջին շերտը կլինի ցանցի ելքը։ Չորս և ավելի շերտ ունեցող նեյրոնային ցանցերին անվանում են նաև խորը նեյրոնային ցանցեր։ Սահմանվում են տարբեր տեսակի շերտեր, որոնք միմյանցից տարբերվում են տվյալ շերտի նեյրոնները նախորդ շերտի նեյրոններին միացնելու ձևով։ Հարկ է նշել նաև, որ սովորաբար մեկ շերտում բոլոր նեյրոնների ակտիվացիաների ֆունկցիաները լինում են նույնը, հետևաբար հաճախ ակտիվացիայի ֆունկցիայի գաղափարը կվերագրենք շերտին։ Շերտի նեյրոնների ակտիվացիաներից կազմված զանգվածը կանվանենք շերտի ելք։

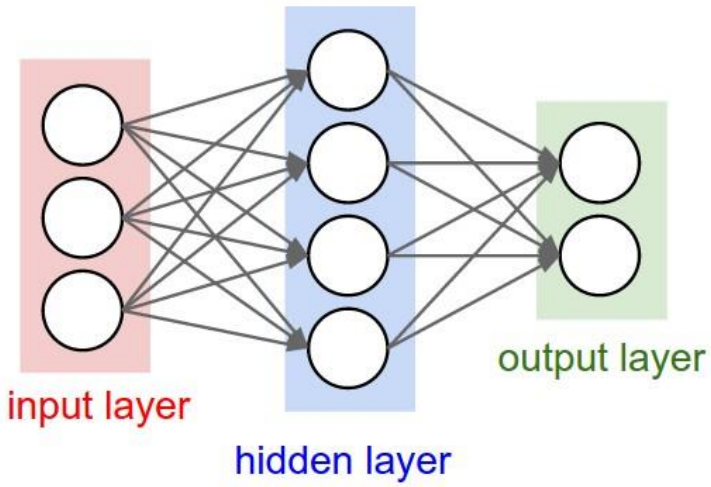
Բազմաշերտ նեյրոնային ցանցերի ուսուցումը կատարվում է Backpropagation ալգորիթմի միջոցով [6]։ Հարկ է նշել, որ բավականաչափ նեյրոններ ունենալու և ոչ գծային ակտիվացիայի ֆունկցիաներ օգտագործելու դեպքում, նեյրոնային ցանցը կարող է ցանկացած ճշտությամբ մոտարկել կամայական ֆունկցիա [7]։

### 1.3.1 Լրիվ կապակցված շերտ

Եթե $y$-ով նշանակենք լրիվ կապակցված շերտի (fully connected layer) ելքի վեկտորը, իսկ $x$ -ով՝ նախորդ շերտի ելքի վեկտորը, ապա լրիվ կապակցված շերտում $y$-ի հաշվարկը կատարվում է հետևյալ կերպ՝

$$y = f(Wx + b) \quad (y \in R^n, \qquad x \in R^m, \qquad W \in R^{n \times m}, \qquad b \in R^n)$$

$W$-ն և $b$-ն շերտի պարամետրերն են, $b$ -ին անվանում են շեղում (bias): $f$-ը շերտի ակտիվացիայի ֆունկցիան է: Շերտում նեյրոնների քանակը՝ $n$-ը, հանդիսանում է կապակցված շերտի հիպերպարամետր: Լրիվ կապակցված շերտը կարելի է պատկերել հետևյալ կերպ՝




Երկու լրիվ կապակցված շերտ ունեցող նեյրոնային ցանց: Նկարը վերցված է [20]-ից:

### 1.3.2 Փաթույթային շերտ

Փաթույթային շերտում (convolutional layer) նեյրոնները դասավորվում են եռաչափ կառուցվածքով՝ եռաչափ զանգվածի վանդակների պես, ուստի կամայական նեյրոն կարելի է ներկայացնել 3 ինդեքսով՝ $k,\ i,\ j$: Փաթույթային շերտը ենթադրում է, որ իրեն նախորդող շերտի ելքը ևս պետք է ունենա նմանատիպ կառուցվածք, բայց, հնարավոր է, տարբեր

չափսերով: Փաթույթային շերտի ելքը նշանակենք $Y$-ով, իսկ նախորդ շերտի ելքը նշանակենք $X$-ով: Փաթույթային շերտի ելքի հաշվարկը տեղի է ունենում հետևյալ կերպ՝

$$Y_j = f\left(\sum_{i=1}^{k_x} X_i * K_{ij} + b_j\right)$$

$(X \in R^{c_x \text{ x } w_x \text{ x } h_x},\ Y \in R^{c_y \text{ x } (w_x - w + 1) \text{ x } (h_x - h + 1)},\ K_{ij} \in R^{w \text{ x } h},$

$i = 1..c_x,\ j = 1..c_y,\ b_j \in R^{c_y})$

$f$-ը շերտի ակտիվացիայի ֆունկցիան է: $*$-ը դիսկրետ փաթույթ գործողությունն է (discrete convolution): $c_y, w, h$ թվերը փաթույթային շերտի հիպերպարամետրերն են:

$W_{u,v,i,j} = K_{u,v,w+1-i,h+1-j}$ – կերպով ստացվող քառաչափ զանգվածը և $b$ վեկտորը կլինեն փաթույթային ցանցի պարամետրերը: $K_{u,v}$ մատրիցներին անվանում են փաթույթի միջուկներ:

Դիսկրետ փաթույթ գործողության օպերանդները մատրիցներ են, գործողությունը իրականացվում է հետևյալ կերպ՝

$$(X * K)_{ij} = \sum_{p=0}^{w-1} \sum_{q=0}^{h-1} x_{i+p,j+q}\ k_{w-p,h-q}$$

$$(X \in R^{n \text{ x } m},\ K \in R^{w \text{ x } h},\ (X * K) \in R^{(n-w+1)\text{x}(m-h+1)},\ w \le n, h \le m)$$

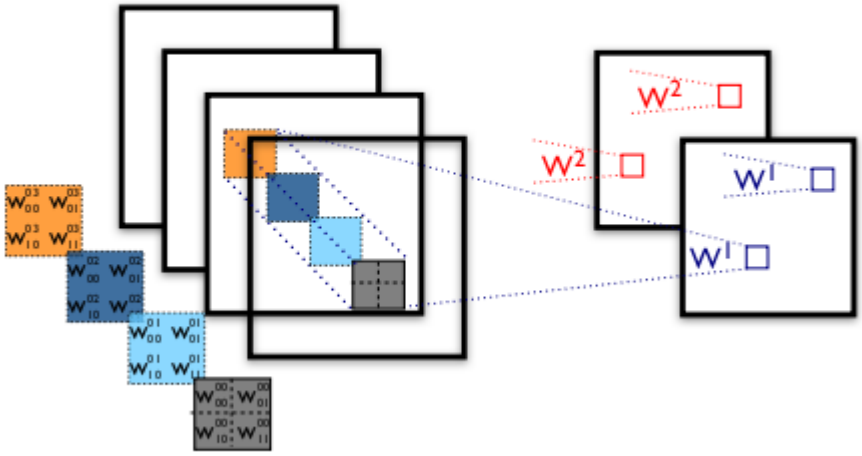
$X_i$ և $K_{ij}$ մատրիցների միջև դիսկրետ փաթույթ գործողությունը հավասար է $X_i$ և $W_{ij}$ մատրիցների միջև դիսկրետ կորելացիա գործողությանը, որը սահմանվում է հետևյալ կերպ՝

$$(X \star W)_{ij} = \sum_{p=0}^{w-1} \sum_{q=0}^{h-1} x_{i+p,j+q}\ W_{p,q}$$

$$(X \in R^{n \text{ x } m},\ W \in R^{w \text{ x } h},\ (X \star W) \in R^{(n-w+1)\text{x}(m-h+1)},\ w \le n, h \le m)$$

Փաթույթային շերտում ստացվում է, որ նեյրոնները բաժանված են խմբերի (խմբի ինդեքսը $k$-ն է), և ամեն խմբում նեյրոնները դասավորված են երկչափ զանգվածում: Յուրաքանչյուր

նեյրոն ինֆորմացիա է ստանում նախորդ շերտի բոլոր խմբերից, բայց ամեն խմբում կապված է միայն $w$ x $h$ չափի լոկալ տիրույթի նեյրոններին: Նույն խմբի մեջ գտնվող նեյրոնների պարամետրերը նույնն են (այս երևույթը կոչվում է parameter sharing): Փաթույթային շերտը կարելի է պատկերել հետևյալ կերպ՝




Վերոնշյալ խմբերը նկարում պատկերված են սև եզրագծերով մեծ ուղղանկյունների տեսքով, նեյրոնները պատկերված են փոքր քառակուսիների տեսքով: $w^1 - ը$ 4 մատրից պարունակող եռաչափ զանգված է: Այդ 4 մատրիցները պատկերված են նկարի ձախ մասում: Ինչպես ցույց է տրված նկարում՝ նույն խմբում գտնվող նեյրոնների պարամետրերը նույնն են. բոլոր կապույտ նեյրոնները ունեն $w^1$ պարամետրերը, իսկ բոլոր կարմիրները՝ $w^2$: Նկարը վերցված է [21]-ից:

### 1.3.3 Ձուլման շերտ

Ձուլման շերտը (pooling layer) սովորաբար դրվում է փաթույթային շերտից հետո: Եթե $X$-ով նշանակենք փաթույթային շերտի ելքը, իսկ $Y$-ով ձուլման շերտի ելքը, ապա $Y$ -ի հաշվարկը տեղի է ունենում հետևյալ կերպ՝

$$Y_{k,i,j} = \max_{0 \le p < p_w,\ 0 \le q < p_h} X_{k,(i-1)s_w+p,(j-1)s_h+q} \quad \text{(max pooling layer)}$$

Կամ

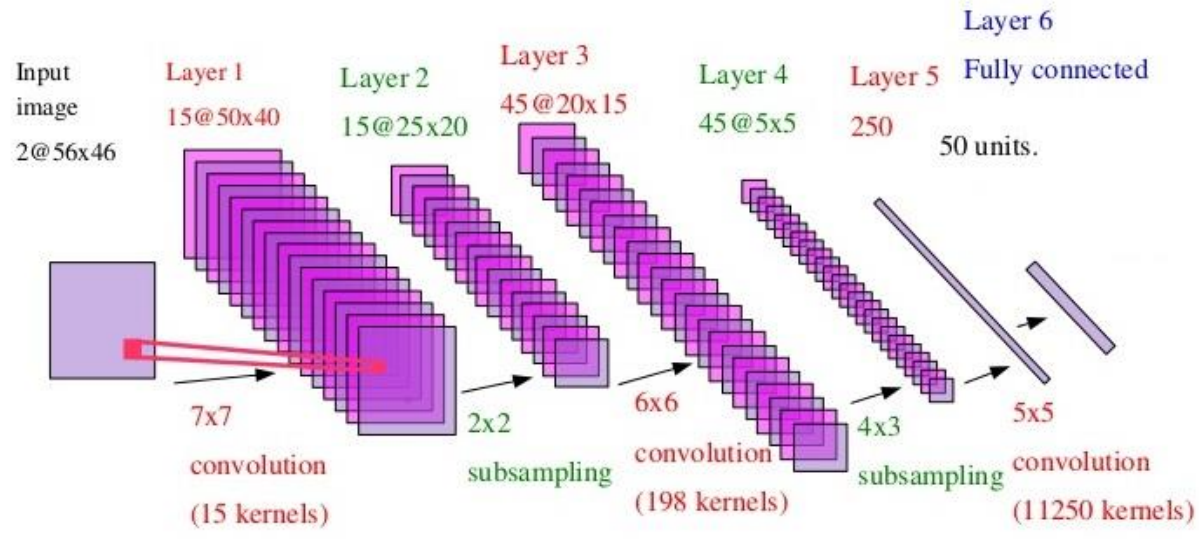
$$Y_{k,i,j} = \frac{1}{p_w p_h} \operatorname{sum}_{0 \le p < p_w,\ 0 \le q < p_h} X_{k,(i-1)s_w+p,(j-1)s_h+q} \quad \text{(average pooling layer)}$$

$X \in R^{n \text{ x } m}$, $Y_{k,i,j} \in R^{u \text{ x } v}$, որտեղ $u$-ն այն ամենամեծ բնական թիվն է, որ $(u-1)s_w + p_w - 1 \le n$, և $v$-ն այն ամենամեծ բնական թիվն է, որ $(v-1)s_h + p_h - 1 \le m$:

$p_w$ և $p_h$ թվերին անվանում են ձուլման տիրույթի չափեր (pooling region size), $s_w$ և $s_h$ թվերին՝ տեղափոխման քայլի չափեր խմբի առաջին և երկորդ առանցքներով (stride size): Ձուլման շերտը պարամետրեր չունի: Շերտը ունի 4 հիպերպարամետրեր՝ $s_w, p_w, s_h, p_h$:

## 1.4 Փաթույթային նեյրոնային ցանց

Փաթույթային անվանում են այն բազմաշերտ նեյրոնային ցանցերին, որոնց հիմնական բաղադրիչը փաթույթային շերտերն են: Փաթույթային ցանցերի առաջին նշանակալի օրինակը LeNet-5 [8] ցանցն էր, որը կատարում էր ձեռագիր թվանշանների դասակարգում հանրահայտ MNIST ձեռագիր թվանշանների տվյալների բազայի վրա:




Նկարում պատկերված է 6 շերտ ունեցող LeNet փաթույթային նեյրոնային ցանց, որը ստանալով մուտքում 2 56x46 չափի անգույն նկարներ, ելքում ստանում է 50 երկարությամբ վեկտոր:

## 1.5 Ռեկուրենտ նեյրոնային ցանց

Ռեկուրենտ նեյրոնային ցանցերը աշխատում են հաջորդական տվյալների հետ: Այսպիսի ցանցերի մուտքային տվյալները, ի տարբերություն ուղիղ տարածող ցանցերի, պարտադիր չէ, որ ունենան ֆիքսված երկարություն: Ռեկուրենտ նեյրոնային ցանցերի մուտքային տվյալները հաջորդականություններ են, որոնց անդամները $n$ չափանի վեկտորներ են: $t$-րդ պահին ցանցի մուտքը հաջորդականության $t$-րդ անդամն է:

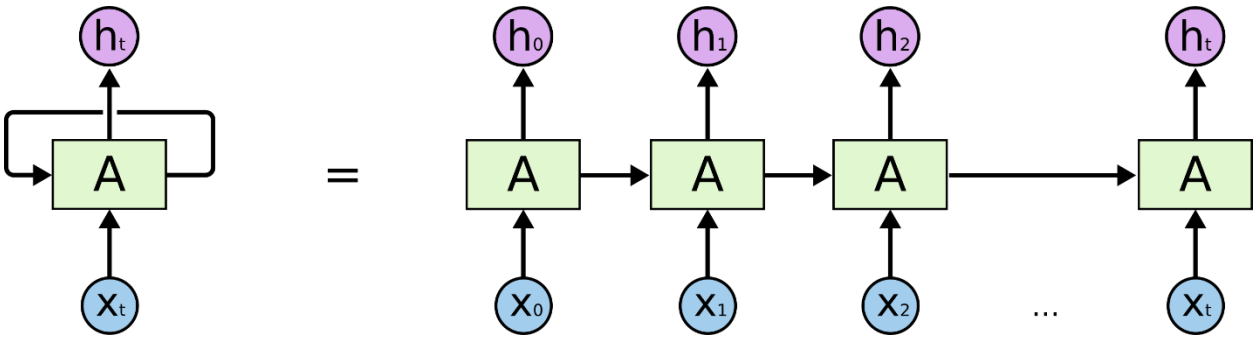
Ռեկուրենտ նեյրոնային ցանցերում ելքը որոշելու համար նախ որոշվում են ելքային նեյրոնները: Ժամանակի $t$-րդ պահին ցանցի ելքը ելքային նեյրոնների t-րդ պահի ակտիվացիաների միավորումից ստացված վեկտորն է: Ռեկուրենտ ցանցի ելքը հաջորդականություն է, որի $t$-րդ անդամը $t$-րդ պահին ցանցի ելքն է:

Ռեկուրենտ նեյրոնային ցանցերին սովորեցնում են Backpropagation through time (BTT) ալգորիթմի միջոցով: Սովորական ռեկուրենտ ցանց կանվանենք հետևյալ հաշվարկներ կատարող ցանցը՝

$$h_t = f(Uh_{t-1} + Wx_t + b)$$

$$(h_t \in R^m,\ x_t \in R^n,\ U \in R^{m \times m},\ W \in R^{m \times n},\ b \in R^m,\ t = 1..T,\ h_0 = 0)$$

$h_t$ վեկտորը կոչվում է ցանցի ներքին վիճակի վեկտոր (hidden state): $W$-ն և $b$-ն ցանցի պարամետրերն են: Այս ցանցը ռեկուրենտ նեյրոնային ցանցերից պարզագույնն է, այն կարելի է պատկերել 2 եղանակով՝




Ձախ մասում ցանցը պատկերված է չբացված վիճակում, և բացահայտորեն ցույց է տրված ռեկուրենտ կապը: Աջ մասում նույն ցանցը պատկերված է ժամանակի առանցքով բացված վիճակում: Նկարը վերցված է [22]-ից:

### 1.5.1 GRU ռեկուրենտ նեյրոնային ցանց

Սովորական ռեկուրենտ ցանցերի BTT ալգորիթմով ուսուցման ընթացքում առաջ են գալիս խնդիրներ, որոնք անհնարին են դարձնում սովորական ռեկուրենտ ցանցերի կիրառումը երկար հաջորդականությունների վրա: Այնպես է ստացվում, որ սովորական ռեկուրենտ նեյրոնային ցանցում ժամանակի $t_2$-րդ պահի ելքի ածանցյալը ըստ $t_1$-րդ ($t_1 \ll t_2$) պահի ելքի կա՛մ ձգտում է 0-ի, կա՛մ ձգտում է անվերջության: Առաջին դեպքում ցանցը չի կարողանում $t_1$-րդ պահին ստացած ինֆորմացիան պահպանել մինչև $t_2$-րդ

պահը և հետևաբար չի սովորում ժամանակի առանցքով երկար կախվածությունները, իսկ երկրորդ դեպքում BTT ալգորիթմի աշխատանքը պարզապես խափանվում է, քանի որ բոլոր ակտիվացիաները և ածանցյալները շատ արագ ընդունում են մեծ արժեքներ: Այս ամենին մանրամասն կարելի է ծանոթանալ [9]-ում:

Վերոնշյալ խնդիրը հաղթահարելու համար մշակվել են մի քանի նոր ռեկուրենտ նեյրոնային ցանցեր, որոնք ունեն ավելի «երկար հիշողություն»: Այդպիսիք են LSTM և GRU ցանցերը [10][11]: Այս աշխատանքում կիրառվել են միայն GRU ռեկուրենտ նեյրոնային ցանցերը, քանի որ դրանք ավելի արագագործ են և դրանցով ստացված արդյունքները չեն զիջում LSTM-ով ստացված արդյուքներին: GRU և LSTM ցանցերի հիմնական տարբերությունը սովորական ռեկուրենտ նեյրոնային ցանցերի նկատմամբ այն է, որ ցանցի ներքին վիճակի վեկտորը շրջափակված է անցախուցերով, որոնք վերահսկում են ինֆորմացիայի մուտքն ու ելքը: GRU ցանցում հաշվարկները կատարվում են հետևյալ կերպ՝

$$z_t = sigmoid(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)})$$

$$r_t = sigmoid(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)})$$

$$\widetilde{h_t} = tanh(Wx_t + Uh_{t-1} + b^{(h)})$$

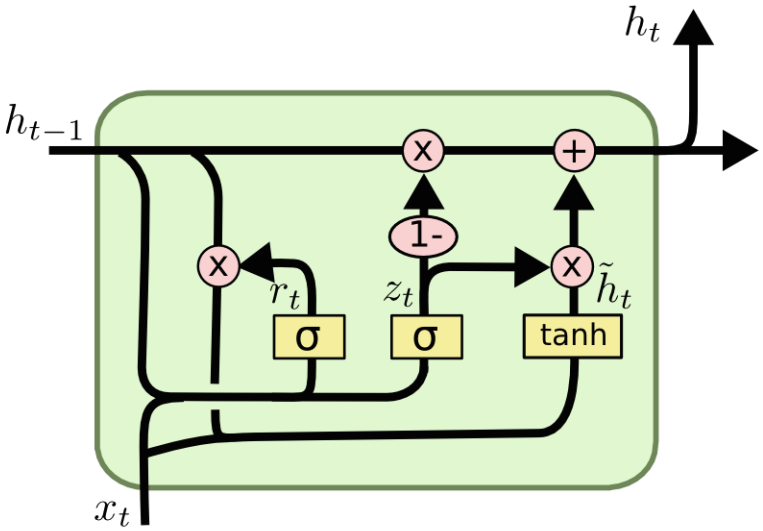
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \widetilde{h_t}$$

$$(h_t \in R^m,\ x_t \in R^n,\ U, U^{(z)}, U^{(r)} \in R^{m \times m},\ W, W^{(z)}, W^{(r)} \in R^{m \times n},$$

$$b^{(z)}, b^{(r)}, b^{(h)} \in R^m,\ t = 1..T,\ h_0 = 0)$$

«∘» գործողությունը վեկտորների անդամ առ անդամ բազմապատկման գործողությունն է: $U, U^{(z)}, U^{(r)},\ W, W^{(z)}, W^{(r)}$ մատրիցները և $b^{(z)}, b^{(r)}, b^{(h)}$ վեկտորները ցանցի պարամետրերն են: Անցախուցերը երկուսն են՝ z և r վեկտորները:

GRU ռեկուրենտ ցանցի մեկ քայլը կարող ենք պատկերել հետևյալ կերպ՝




Սլաքներով ցույց են տրված ինֆորմացիայի հոսքերը, ուղղանկյուններով՝ ակտիվացիայի ֆունկցիաներ կիրառման տեղերն են, շրջաններով՝ թվաբանական գործողությունների կիրառման տեղերը։ Նկարը վերցված է [22]-ից։

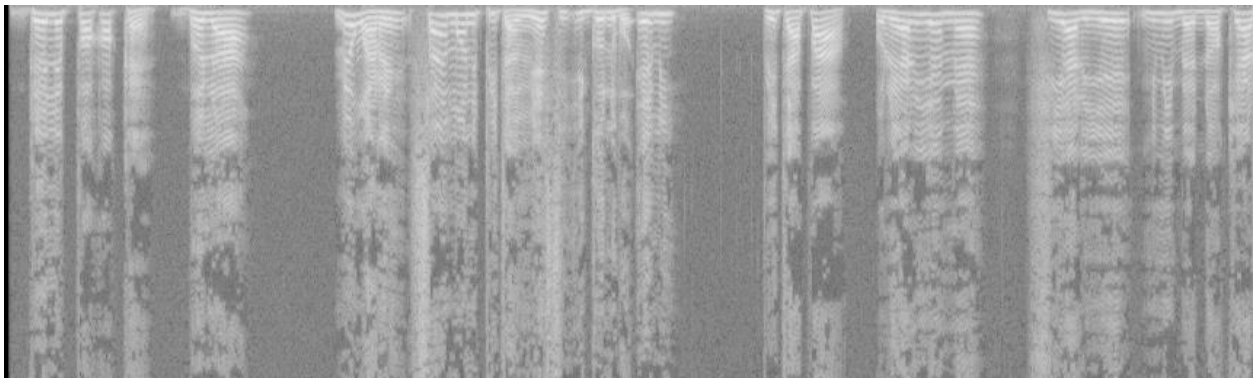
### 1.5.2 Ռեկուրենտ նեյրոնային ցանցը որպես բազմաշերտ նեյրոնային ցանցի շերտ

Ռեկուրենտ նեյրոնային ցանցը կարելի է դիտարկել որպես բազմաշերտ նեյրոնային ցանցի շերտ։ Այն մուտքում ստանում է երկչափ զանգված և վերադարձնում երկչափ զանգված։ Մուտքի $t$-րդ տողը դիտարկում ենք որպես ռեկուրենտ նեյրոնային ցանցի ժամանակի $t$-րդ պահի մուտք, իսկ ելքի $t$-րդ տողը կլինի ցանցի $t$-րդ պահի ելքը։ Այս աշխատանքում օգտագործվել է ռեկուրենտ նեյրոնային ցանցերի հենց այս մեկնաբանությունը, որը հնարավորություն է տալիս կառուցել բազմաշերտ ռեկուրենտ նեյրոնային ցանցեր և օգտագործել ռեկուրենտ շերտեր փաթույթային նեյրոնային ցանցերում։

# 2 Խնդիր և տվյալներ

Խոսքից լեզվի ճանաչման խնդիրը առաջարկվել է TopCoder կազմակերպության կողմից։ Տրված են 67176 հատ 10 վայրկյան տևողությամբ խոսքի օրինակներ, որոնցից յուրաքանչյուրի համար նշված է թե որ լեզուն է հնչում, և ևս 12320 հատ օրինակ, որոնց համար նշված չէ լեզուն. այն պետք է կանխատեսել։ Բոլոր օրինակներում աղմուկ գրեթե չկա, և հնչում է մեկ մարդու ձայն։ Ձայնագրությունները 176 տարբեր լեզուներով են, որոնք համեմատաբար քիչ հայտնի լեզուներ են (հիմնականում Աֆրիկայի, Ասիայի և Օվկիանիայի ժողովուրդների լեզուներ)։

Բոլոր օրինակները տրված են *mp3* ֆորմատով, որը մենք փոխարկում ենք *wav* ֆորմատի։ *Wav* ֆորմատում պահվում է ձայնի ալիքի դիսկրետ տարբերակը՝ 44100 Հց հաճախականությամբ։ Տասը վայրկյան տևողությամբ խոսքի ձայնագրության հաջորդականությունը պարունակում է 441000 թիվ։ Այդպիսի հաջորդականությունից նեյրոնային ցանցի համար դեռևս դժվար է գտնել հնչող լեզուն, անհրաժեշտ է տվյալների նոր ներկայացում։ Այդ պատճառով ամեն օրինակի համար կառուցում ենք ձայնագրության սպեկտրոգրամը։ Ստացվում են 256x858 չափի այսպիսի նկարներ՝



Նկարի հորիզոնական առանցքը ժամանակն է, ուղղահայաց առանցքը՝ հաճախականությունը, ընդ որում նկարի վերևում ավելի ցածր հաճախականություններն են։ Ժամանակի ցանկացած պահին համապատասխան սյան մեջ գրված են այդ պահին ձայնի ալիքի Ֆուրյեի շարքի գործակիցները (գործակիցների մեծացման ուղղությունը սևից սպիտակն է)։ Մաքսիմալ հաճախականությունը 11 կՀց է։

Տվյալների ներկայացման այս ձևը բավականին հարմար է խորը փաթույթային նեյրոնային ցանցերի համար։ Ցանցը մուտքում ստանում է $w \times h$ չափի $[0,1]$ միջակայքի թվերից

կազմված մատրից և վերջին շերտում վերադարձնում 176 չափանի հավանականությունների վեկտոր։ Այս աշխատանքում օգտագործվում է նկարի միայն վերևի կեսը՝ որը համապատասխանում է հաճախականությունների [0 – 5.5 կհց] միջակայքին։ Մարդու խոսքի հարմոնիկները հիմնականում հենց այդ միջակայքում է լինում։

# 3 Մոտեցումներ

Այս աշխատանքում օգտագործել ենք 3 տիպի նեյրոնային ցանց՝

- Փաթույթային նեյրոնային ցանց
- GRU ռեկուրենտ նեյրոնային ցանց
- Նախորդ երկուսի համադրումից ստացված ցանց

Ստորև կներկայացնենք ամեն տիպից ամենահաջողված մոդելի նկարագրությունները։

## 3.1 Փաթույթային նեյրոնային ցանց

| | Տիպ | Խումբ | Լայնություն | Երկարություն | Միջուկի չափ / քայլ | Ակտիվացիայի ֆունկցիա | |
|---|---|---|---|---|---|---|---|
| 0 | Input | 1 | 128 | 858 | | | |
| 1 | Conv | 16 | 122 | 852 | 7x7 / 1 | Relu | |
| 2 | MaxPool | 16 | 62 | 427 | 3x3 / 2 | | Pad=2 |
| 3 | Batch Norm | 16 | 62 | 427 | | | |
| 4 | Conv | 32 | 58 | 423 | 5x5 / 1 | Relu | |
| 5 | MaxPool | 32 | 30 | 213 | 3x3 / 2 | | Pad=2 |
| 6 | Batch Norm | 32 | 30 | 213 | | | |
| 7 | Conv | 32 | 28 | 211 | 3x3 / 1 | Relu | |
| 8 | MaxPool | 32 | 15 | 107 | 3x3 / 2 | | Pad=2 |
| 9 | Batch Norm | 32 | 15 | 107 | | | |
| 10 | Conv | 64 | 13 | 105 | 3x3 / 1 | Relu | |
| 11 | MaxPool | 64 | 8 | 54 | 3x3 / 2 | | Pad=2 |
| 12 | Batch Norm | 64 | 8 | 54 | | | |
| 13 | Conv | 64 | 6 | 52 | 3x3 / 1 | Relu | |
| 14 | MaxPool | 64 | 4 | 27 | 3x3 / 2 | | Pad=2 |
| 15 | Batch Norm | 64 | 4 | 27 | | | |
| 16 | Fully connected | 256 | | | | Relu | |
| 17 | Batch Norm | 256 | | | | | |
| 18 | Dropout | 256 | | | | | Prob=0.5 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 19 | Fully connected | 176 | | | | Softmax | |

«Batch Norm» և «Dropout» շերտերը մեկնաբանվում են համապատասխանաբար 4-րդ և 5-րդ բաժիններում:

## 3.2 GRU ռեկուրենտ նեյրոնային ցանց

| | **Տիպ** | **Մուտքի երկարություն** | **Ներքին վիճակի նեյտրոններ** | **Ակտիվացիայի ֆունկցիա** | |
|---|---|---|---|---|---|
| 0 | Input | 858 | 128 | | |
| 1 | GRU | 858 | 500 | Tanh | |
| 2 | Batch Norm | 858 | 500 | | |
| 3 | GRU | | 500 | Tanh | Միայն վերջին ելքը |
| 4 | Batch Norm | | 500 | | |
| 5 | Fully connected | | 176 | Softmax | |

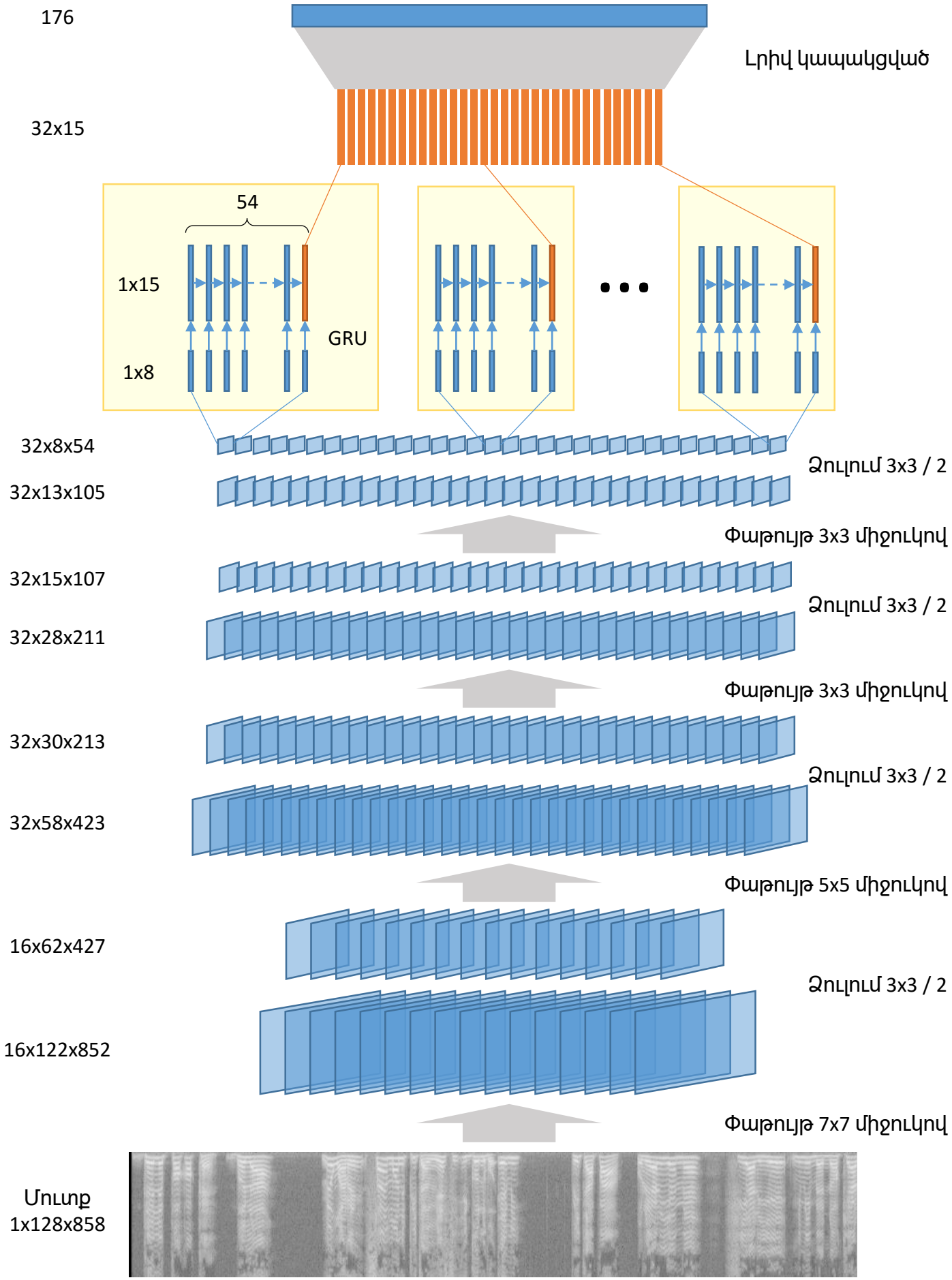
«Batch Norm» և «Dropout» շերտերը մեկնաբանվում են համապատասխանաբար 4-րդ և 5-րդ բաժիններում:

## 3.3 Փաթույթային + ռեկուրենտ ցանց

| | Տիպ | Խումբ | Լայնություն | Երկարություն | Միջուկի չափ / քայլ | Ակտիվացիայի ֆունկցիա | |
|---|---|---|---|---|---|---|---|
| 0 | Input | 1 | 128 | 858 | | | |
| 1 | Conv | 16 | 122 | 852 | 7x7 / 1 | Relu | |
| 2 | MaxPool | 16 | 62 | 427 | 3x3 / 2 | | Pad=2 |
| 3 | Batch Norm | 16 | 62 | 427 | | | |
| 4 | Conv | 32 | 58 | 423 | 5x5 / 1 | Relu | |
| 5 | MaxPool | 32 | 30 | 213 | 3x3 / 2 | | Pad=2 |
| 6 | Batch Norm | 32 | 30 | 213 | | | |
| 7 | Conv | 32 | 28 | 211 | 3x3 / 1 | Relu | |
| 8 | MaxPool | 32 | 15 | 107 | 3x3 / 2 | | Pad=2 |
| 9 | Batch Norm | 32 | 15 | 107 | | | |
| 10 | Conv | 32 | 13 | 105 | 3x3 / 1 | Relu | |
| 11 | MaxPool | 32 | 8 | 54 | 3x3 / 2 | | Pad=2 |
| 12 | Batch Norm | 32 | 8 | 54 | | | |
| 13 | GRU | 32 | 15 | 1 | 3x3 / 1 | Tanh | Միայն վերջին ելքը: |
| 14 | Concatenation | 480 | | | | | |
| 15 | Batch Norm | 480 | | | | | |
| 16 | Fully connected | 176 | | | | Softmax | |

«Batch Norm» շերտը մեկնաբանվում է 5րդ բաժնում:

Այս ցանցում կարևոր է 13-րդ շերտը, որում գործում են 32 հատ նույն պարամետրերով GRU ռեկուրենտ նեյրոնային ցանցեր: Յուրաքանչյուր GRU գործում է 54x8 չափ ունեցող հաջորդական տվյալների վրա և վերադարձնում 15 չափանի վեկտոր: 14-րդ շերտում այդպիսի 32 հատ վեկտորները կցվում են իրար, և ստացվում է 480 չափանի վեկտոր: Ցանցը պատկերված է հաջորդ էջում:


176
Լրիվ կապակցված
32x15
54
1x15
1x8
GRU
32x8x54
Զուլում 3x3 / 2
32x13x105
Փաթույթ 3x3 միջուկով
32x15x107
Զուլում 3x3 / 2
32x28x211
Փաթույթ 3x3 միջուկով
32x30x213
Զուլում 3x3 / 2
32x58x423
Փաթույթ 5x5 միջուկով
16x62x427
Զուլում 3x3 / 2
16x122x852
Փաթույթ 7x7 միջուկով
Մուտք
1x128x858

# 4 Օպտիմիզացիա

Նախորդ բաժնում թվարկված բոլոր մոդելների վերջին շերտում ստացվում է $(\hat{p}_1, \hat{p}_2, \dots, \hat{p}_{176})$ հավանականությունների վեկտոր: $\hat{p}_i$-ն ցույց է տալիս հավանականությունը, որ տվյալ օրինակի մեջ հնչում է $i$-րդ լեզուն: Հնչող լեզվի համար ցանցի վերջնական գնահատականը կլինի՝ $\hat{y} = \underset{i=1..176}{\operatorname{argmax}} \hat{p}_i$ :

Տվյալներում տրված է տվյալ օրինակի համար հնչող լեզվի համարը՝ $y$: Այս փաստը ևս կարող ենք տալ հավանականություններ վեկտորով՝ $p_i = 1_{[i=y]}$: Ընտրենք կորստի ֆունկցիան այնպես, որ այն գնահատի $p$ վեկտորը $\hat{p}$ -ով մոտարկման սխալանքը: Այս աշխատանքում ամենուրեք որպես կորստի ֆունկցիա ընտրել ենք դիսկրետ փոխադարձ էնթրոպիայի կորստի ֆունկցիան (discrete cross entropy loss function):

$$L(p, \hat{p}) = -\sum_{i=1}^{176} p_i \log(\hat{p}_i)$$

Օպտիմիզիացիայի խնդիրը կլինի հետևյալը՝

$$\tilde{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L\big(p, \hat{p}(x_i)\big) \to min$$

($\theta$-ն ցանցի բոլոր պարամետրերից կազմված վեկտորն է, $\theta \in R^P$)

Խորը ուսուցման մեջ այսպիսի օպտիմիզացիայի խնդիրների համար հիմնականում օգտագործում են գրադիենտային անկման տարատեսակներ: Սովորական գրադիենտային անկման ժամանակ ամեն իտերացիայի ժամանակ կատարվում է հետևյալը՝

$\theta_t = \theta_{t-1} - \alpha \nabla_\theta \left(\tilde{L}(\theta)\right)$ $(*)$, որտեղ $\alpha$-ն ալգորիթմի միակ հիպերպարամետրն է և կոչվում է սովորելու քայլ (learning rate):

Այս ալգորիթմը կիրառելի չէ մեր դեպքում , քանի որ $\nabla_\theta(\tilde{L}(\theta))$-ը հաշվելու համար պետք է ցանցը աշխատեցնել բոլոր օրինակների վրա և նոր կատարել պարամետրերի փոփոխությունը, իսկ ցանցը ամբողջ տվյալների վրա աշխատեցնելը տևում է 10 րոպեից մինչև մեկ ժամ: Այս խնդիրը լուծելու համար ընդունված է կատարել $(*)$-ը ամեն $b$-րդ

օրինակից հետո՝ միջինացնելով $b$ հատ օրինակների կորստի ֆունկցիաների գրադիենտները: Այս աշխատանքի բոլոր մոդելներում ընտրել ենք $b = 32$:

Օպտիմիզացիայի արագությունը և որակը բարձրացնելու համար գոյություն ունեն գրադիենտային անկման մի շարք տարատեսակներ: Այս աշխատանքում օգտագործվել են դրանցից երկուսը՝ momentum [13] և adadelta [12]:

Momentum-ի դեպքում ամեն իտերացիայում կատարվում են հետևյալ գործողությունները՝

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \tilde{L}(\theta)$$
$$\theta_t = \theta_{t-1} - v_t$$

Այս աշխատանքում $\gamma = 0.9$:

Adadelta-ի դեպքում ամեն իտերացիայում կատարվում են հետևյալ գործողությունները՝

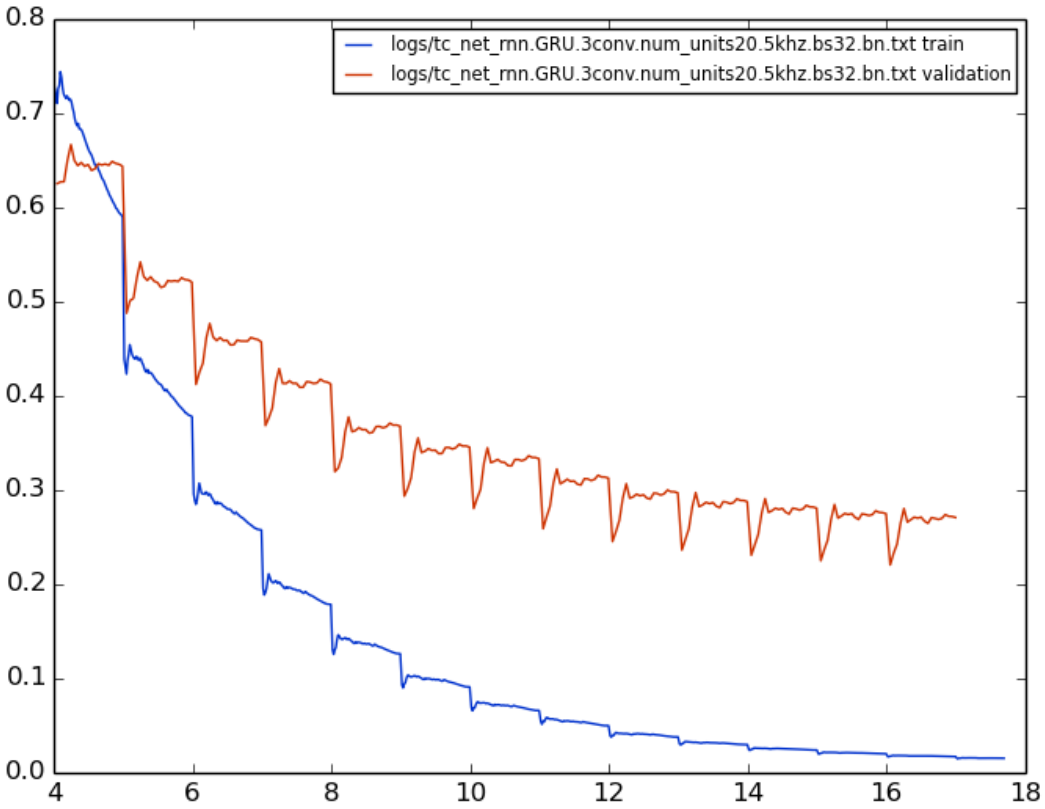
$$g_t = \nabla_\theta \tilde{L}(\theta)$$
$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$$
$$\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t$$
$$E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$$
$$\theta^{(t)} = \theta^{(t-1)} + \Delta\theta_t$$

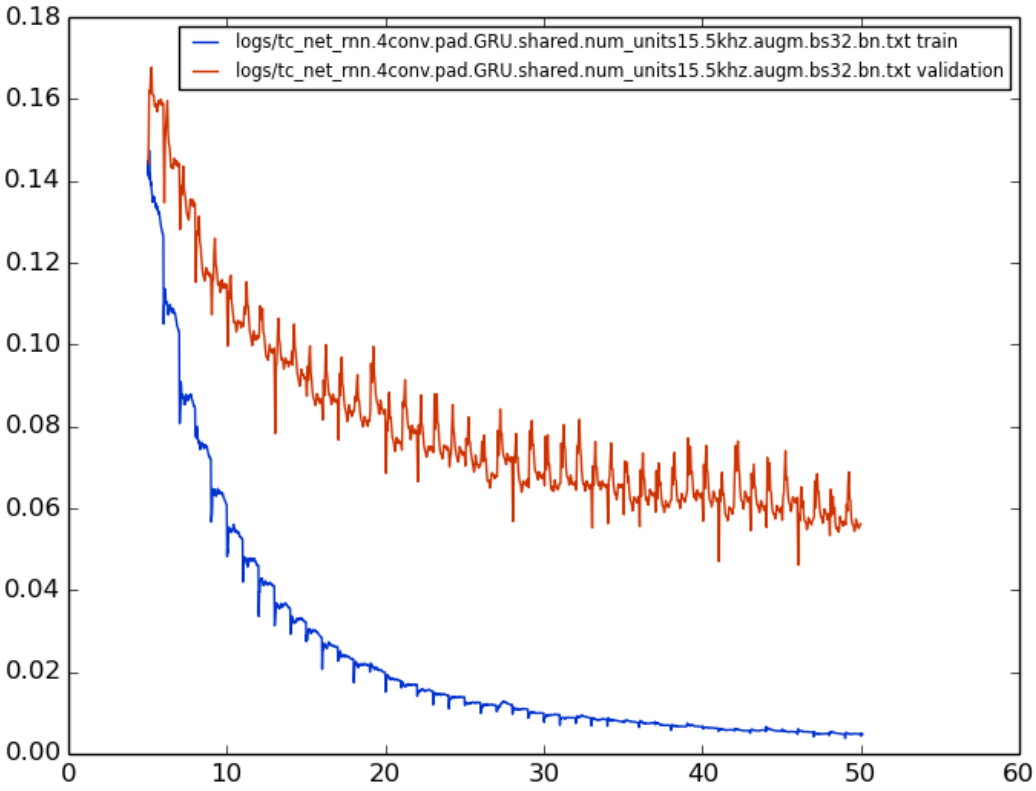
Այս աշխատանքում $\rho = 0.95$ և $\epsilon = 10^{-6}$:

Օպտիմիզացիայի արագությունը բարձրացնելու նպատակով օգտագործել ենք Batch normalization [14] տիպի շերտ, որի նպատակն է նորմալիզացնել բազմաշերտ նեյրոնային ցանցի շերտերի ելքերը, որպեսզի ելքերը միշտ լինեն նույն բաշխումից:

# 5 Գերմոտարկում

Մեքենայական ուսուցման մեջ ընդհանուր խնդիր է գերմոտարկման (ovefitting) խնդիրը: Գերմոտարկման դեպքում ցանցը լավ մոտարկում է ֆունկցիան այն կետերում, որոնք օգտագործվել են ցանցին սովորեցնելու ժամանակ և համեմատաբար ավելի վատ է մոտարկում նոր օրինակների համար: Այս աշխատանքում օգտագործված գրեթե բոլոր ցանցերը 100% ճշտության են հասնում այն օրինակների վրա, որոնք օգտագործվել են ցանցին սովորեցնելու ժամանակ: Գերմոտարկումը շատ լավ երևում է, երբ նկարում ենք կորստի ֆունկցիայի իտերացիաների քանակից ունեցած կախման գրաֆիկը՝






Գրաֆիկներում պատկերված են կորստի ֆունկցիայի արժեքները ցանցի ուսուցման ընթացքում։ Կապույտով պատկերված է կորստի ֆունկցիայի արժեքը այն օրինակների վրա, որոնք օգտագործվել են ցանցին սովորեցնելու համար, իսկ կարմիրով՝ կորստի ֆունկցիան նոր օրինակների վրա։ Առաջին ցանցում գերմոտարկումը բավականին բարձր է՝ կապույտ և կարմիր գրաֆիկները տարբերությունը մոտ 0.3 է։ Երկրորդ ցանցում գերմոտարկում կա, բայց այն համեմատաբար ավելի քիչ է, քանի որ կապույտ և կարմիր գրաֆիկները տարբերությունը մոտ 0.06 է։ Նշենք, որ առաջին ցանցի ճշտությունը օգտագործված և նոր օրինակների վրա համապատասխանաբար 100% և 92.4% է, իսկ երկրորդ ցանցի համար այդ ցուցանիշները համապատասխանաբար 100% և 98.45% են։

Գերմոտարկումը նվազեցնելու համար կիրառել ենք 3 եղանակ՝

1. Տվյալների արհեստական հարստացում
2. Dropout
3. L2 ռեգուլյարիզացիա

## 5.1 Տվյալների արհեստական հարստացում

Տվյալների արհեստական հարստացման ժամանակ տվյալների բազայում եղած օրինակներից որոշակի գործողությունների արդյունքում գեներացվում են նոր օրինակներ, որոնք տարբերվում են սկզբնական օրինակներից բայց չեն փոխում իրենց դասը (մեր դեպքում չի փոխվում լեզուն)։ Այս աշխատանքում կիրառվել է տվյալների հարստացման 2 եղանակ՝

1. 10 վայրկյան տևողությամբ խոսքից պատահականորեն ընտրում ենք 9 վայրկյան տևողությամբ կտոր։ Այս գործողությունը կատարում ենք ուսուցման ընթացքում՝ ավելորդ հիշողություն չզբաղեցնելու համար։ Ձայնագրության համար վերջնական պատասխանը կարելի է ստանալ այդ ձայնագրության $k$ տարբեր 9 վայրկյան տևողությամբ կտորների վրա ցանցի տված հավանականությունները միջինացնելով։ Այս աշխատանքում ընտրել ենք $k = 20$։
2. Սկզբնական շրջանում փաթույթային նեյրոնային ցանցում նախորդ եղանակի հետ միասին օգտագործվել է [15]-ում նկարագրված եղանակը տվյալները արհեստականորեն հարստացնելու համար։ Այն փոքր չափով դեֆորմացիայի է ենթարկում սպեկտրոգրամի հաճախականությունների առանցքը՝ որոշ տեղերում ձգելով այն, որոշ տեղերում՝ սեղմելով։ Թեև այս եղանակը օգնում է լավացնել արդյունքները, սակայն, այն համար չէ կիրառել ուսուցման ընթացքում։ Այս եղանակը օգտագործելու համար պահանջվում է մեծ քանակությամբ հիշողություն, որպեսզի ամեն նկարից ստացված նոր $k$ նկարները պահվեն։

## 5.2 Dropout շերտ

Dropout շերտը [16] հատուկ տիպի շերտ է, որը պատահական սկզբունքով իրեն նախորդող շերտի նեյրոնների ակտիվացիաները դարձնում է 0, կամ թողնում է նույնը։ Նեյրոնի ակտիվացիայի 0 դառնալու հավանականությանը կանվանենք dropout շերտի հավանականություն։ Dropout շերտի հավանականությունը ևս հիպերպարամետր է, որը ընտրել ենք փորձնական ճանապարհով։ Dropout շերտը «ստիպում» է նեյրոններին լինել միմյանցից անկախ։

## 5.3 L2 ռեգուլյարիզացիա

L2 ռեգուլարիզացիայի ժամանակ էմպիրիկ կորստի ֆունկցիային գումարվում է $\theta$ պարամետրի $L2$ նորմը $\lambda$ գործակցով բազմապատկած։ $\lambda$ գործակիցը հիպերպարամետր է, որը պետք է ընտրել փորձնական ճանապարհով։ L2 ռեգուլյարիզացիան «ստիպում» է, որ ցանցի պարամետրերը չլինեն շատ մեծ։ Այդ կերպ ցանցը արդյունքում ստանում է ավելի ողորկ ֆունկցիաներ, որոնք ավելի քիչ են գերմոտարկում։

# 6 Իրականացում

Այս աշխատանքի համար անհրաժեշտ գրեթե բոլոր ծրագրերը գրվել են python ծրագրավորման լեզվով: Օգտագործվել են սիմվոլիկ հաշվարկների և ավտոմատ դիֆերենցման համար նախատեսված Theano [17] գրադարանը և նեյրոնային ցանցերի համար նախատեսված Lasagne [18] գրադարանը: Սկզբնական շրջանում օգտագործվել է նաև նեյրոնային ցանցերի համար նախագծված Caffe [17] գրադարանը, որը հարմար է ստանդարտ մոդելների արագ նախագծման և իրականացման համար: Հետագայում Caffe-ի բավարար ճկունություն չապահովելու պատճառով անցում է կատարվել Theano գրադարանին:

Theano գրադարանը ցանցի ուսուցման ալգորիթմը թարգմանում է Nvidia վիդեոքարտերի համար նախատեսված CUDA լեզվի: Բոլոր հաշվարկները իրականացվել են Nvidia GTX 980 վիդոքարտի վրա: Մեկ մոդելի ուսուցումը միջինում տևում է 5 ժամից մինչև 1 օր : Մեկ ձայնագրության վրա ցանցերը աշխատում են միջինում 0.02 վայրկյան:

# 7 Արդյունքներ

Այս բաժնում կներկայացնենք միայն չօգտագործված օրինակների վրա ցանցերի ցույց տված արդյունքները՝ սխալանքի չափը։

## 7.1 Փաթույթային ցանց

Փաթույթային ցանցով ստացված լավագույն մոդելը ներկայացված է 3.1 բաժնում։ Այդ մոդելը տալիս է **3.5%** սխալանք։ Այդ ցանցից 2 անգամ չափերով ավելի մեծ նույն տիպի ցանցը տալիս է **3.8%** սխալանք։

## 7.2 Ռեկուրենտ ցանց

Ռեկուրենտ ցանցով ստացված լավագույն մոդելը ներկայացված է 3.2 բաժնում։ Այդ մոդելը տալիս է **1.58%** սխալանք։ Գերմոտարկումը նվազեցնելու համար ուսուցման պրոցեսի կեսից ներմուծվել է L2 ռեգուլյարիզացիա՝ $\lambda = 0.001$ գործակցով։ Մեկ շերտ պարունակող ռեկուրենտ ցանցով լավագույն արդյունքը, որ հաջողվել է ստանալ եղել է **8.12%**։

## 7.3 Փաթույթային + ռեկուրենտ ցանց

Ռեկուրենտ և փաթույթային ցանցերի միասին օգտագործումից ստացված լավագույն մոդելը ներկայացված է 3.3 բաժնում։ Այդ մոդելը տալիս է **2%** սխալանք։ Այդ մոդելում օգտագործվել է 4 փաթույթային շերտ։ Երբ օգտագործվում է 3 փաթույթային շերտ, արդյունքները վատանում են, սխալանքը դառնում է **3.04%**։ Երբ մոդելից հանվում է ռեկուրենտ ցանցերի պարամետրերի նույնը լինելու պայմանը, սխալանքը դառնում է **7.6%**։ Բոլոր դեպքերում սխալանքի մեծացումը պատճառ է ոչ թե մոդելի «կարողությունների» փոքրացման, այլ գերմոտարկման մեծացման։ Այս տիպի ցանցերում L2 ռեգուլյարիզացիան և dropout շերտը գրեթե չեն օգնում գերմոտարկումը նվազեցնելու հարցում։ Օգնում է տվյալների արհեստական հարստացման առաջին տարբերակը (5.1.1)։ Այդ եղանակով 3.3 բաժնում նկարագրված մոդելի սխալանքը հաջողվել է հասցնել **1.32%**-ի։

## 7.4 Քվեարկություն

Աշխատանքում ներկայացված բոլոր մոդելների կանխատեսումները կարելի է իրար հետ համատեղել և ստանալ ավելի լավ արդյունքներ։ Այդ նպատակով բոլոր մոդելների կանխատեսած հավանականությունների վեկտորների կցագրումից ստացված վեկտորի վրա կիրառում ենք մեկ լրիվ կապակցված շերտ ունեցող նեյրոնային ցանց, որը տալիս է վերջնական պատասխանը։ Այս եղանակով հաջողվել է ստանալ **0.64%** սխալանք, որը մոտ երկու անգամ ավելի քիչ է, քան առանձին վերցրած լավագույն մոդելի սխալանքը։

# 8 Եզրակացություն

Նեյրոնային ցանցերը, ինչպես և սպասվում էր, մեծ ճշտությամբ գտնում են ձայնագրության մեջ հնչող լեզուն։ Փաթույթային ցանցերը հաջողությամբ մշակում են տվյալները՝ քայլ առ քայլ ստանալով տվյալների համար նոր ներկայացումներ, բայց լավ չեն օգտագործում ժամանակի գործոնը։ Ռեկուրենտ նեյրոնային ցանցերը բացահայտ կերպով հիմնվում են ժամանակի գործոնի վրա և ավելի լավ արդյունքի են հասնում, քան փաթույթային ցանցերը։ Ռեկուրենտ ցանցի խնդիրն այն է, որ ցանցը սկսում է դասակարգում կատարել գրեթե միանգամից սպեկտրոգրամի վրա, որը թեև ձայնագրության բավականին լավ ներկայացում է, սակայն դեռ բավականաչափ մշակված չէ և պարունակում է խնդրի լուծման համար ոչ պետքական ինֆորմացիա։ Հենց այդ պատճառով դիտարկել ենք 3-րդ մոդելը, որը փաթույթային ցանցով մշակում է սպեկտրոգրամը և արդյունքը փոխանցում ռեկուրենտ ցանցին։ Այս մոդելը դեռ շատ փորձարկումների չի ենթարկել, բայց արդեն բավականին լավ արդյունքներ են գրանցվել։ Հետագայում պատրաստվում ենք շարունակել այս մոդելի զարգացումը։

# 9 Գրականություն և հղումներ


[1] Christopher Bishop. Pattern Recognition and Machine Learning. *Springer* , 2006.

[2] Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep Learning. *MIT Press, 2016*.

[3] Gregoire Montavon. Deep learning for spoken language identification. *Microsoft Research, NIPS 2009.*

[4] Wouter Gevaert, Georgi Tsenov, Valeri Mladenov. Neural Networks used for Speech Recognition, *Journal of automatic control, University of Belgrade, Vol. 20:1-7, 2010*.

[5] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS 2012*.

[6] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams. Learning representations by back-propagating errors. *Nature Vol. 323, 1986*.

[7] K. Hornik, M. Stinchcombe, H. White. Dermition. Multilayer feedforward networks are universal approximators. *Neural Networks Vol. 2, pp. 359-366, 1989*.

[8] LeNet-5 convolutional neural networks. URL http://yann.lecun.com/exdb/lenet.

[9] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio. On the difficulty of training recurrent. neural networks. *JMLR Workshop and Conference Proceedings, Vol. 28*, 2013.

[10] Sepp Hochreiter, Jurgen Schmidhuber. Long Short-Term Memory. *Neural Computation 9(8):1735{1780, 1997*.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555*, *2014*.

[12] Matthew Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701, 2012*.

[13] Ilya Sutskever, James Martens, George Dahl, Geofrey Hinton. On the importance of initialization and momentum in deep learning.

[14] Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167, 2015.*

[15] Navdeep Jaitly, Geoffrey E. Hinton. Vocal Tract Length Perturbation (VTLP) improves speech recognition. *ICML 2013*.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, Vol. 15*, *2014*.

[17] Theano. http://deeplearning.net/software/theano

[18] Lasagne. https://github.com/Lasagne/Lasagne

[19] Caffe. http://caffe.berkeleyvision.org

[20] http://cs231n.github.io/neural-networks-1

[21] http://deeplearning.net/tutorial/lenet.html

[22] http://colah.github.io/posts/2015-08-Understanding-LSTMs